SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

AD-A277 507

DTIC ELECTE MAR 3 0 1994 S F D

1a. REPORT SECURITY CLASSIFICATION Unclassified	
2a. SECURITY CLASSIFICATION AUTHORITY	
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	Approved for public release; distribution unlimited.
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) #69
6a. NAME OF PERFORMING ORGANIZATION Institute for Brain and Neural Systems 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Personnel and Training Research Programs Office of Naval Research (Code 1142PT)
6c. ADDRESS (City, State, and ZIP Code) Brown University Providence, Rhode Island 02912	7b. ADDRESS (City, State, and ZIP Code) 800 North Quincy Street Arlington, VA 22217-5000
8a. NAME OF FUNDING / SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-91-J-1316
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11. TITLE (Include Security Classification)

Proceedings of the NIPS'93 Postconference Workshop on Methods for Combining Neural Networks

12. PERSONAL AUTHOR(S)

Michael P. Perrone

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical Report	FROM ____ TO ____	March 15, 1994	51(2-sided)

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Averaging, Ensemble Methods, Committees
05	08		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report is the proceedings from the NIPS'93 postconference workshop entitled "Pulling It All Together: Methods for Combining Neural Networks." The goal of the workshop was to examine theoretical aspects and heuristic methods for combining existing neural network algorithms to generate systems which have improved performance properties.

This report includes an outline and summary of the workshop as well as the slides used in each participant's presentation.

DTIC QUALITY INSPECTED 3

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Dr. Joel Davis	22b. TELEPHONE (Include Area Code) (703) 696-4744 / 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

94 3 29 024

Pulling It All Together: Methods for Combining Neural Networks*

Michael P. Perrone
Institute for Brain and Neural Systems
Brown University
Providence, RI
mpp@cns.brown.edu

[This is a brief summary of the workshop. The overhead slides used by the speakers following this summary.]

The past several years have seen a tremendous growth in the complexity of the recognition, estimation and control tasks expected of neural networks. In solving these tasks, one is faced with a large variety of learning algorithms and a vast selection of possible network architectures. After all the training, how does one know which is the best network? This decision is further complicated by the fact that standard techniques can be severely limited by problems such as over-fitting, data sparsity and local optima. The usual solution to these problems is a *winner-take-all cross-validatory* model selection. However, recent experimental and theoretical work indicates that we can improve performance by considering methods for combining neural networks.

This workshop examined current neural network optimization methods based on combining estimates and task decomposition, including Boosting, Competing Experts, Ensemble Averaging, Metropolis algorithms, Stacked Generalization and Stacked Regression. The issues covered included Bayesian considerations, the role of complexity, the role of cross-validation, incorporation of a priori knowledge, error orthogonality, task decomposition, network selection techniques, overfitting, data sparsity and local optima. Highlights of each talk are given below. To obtain the workshop proceedings, please contact the author or Norma Caccia (*norma_caccia@brown.edu*) and ask for IBNS ONR technical report #69.

M. Perrone (Brown University, "Averaging Methods: Theoretical Issues and Real World Examples") presented weighted averaging schemes [7], discussed their theoretical foundation [6], and showed that averaging can improve performance whenever the cost function is (positive or negative) convex which includes Mean Square Error, a general class of L_p-norm cost functions, Maximum Likelihood Estimation, Maximum Entropy, Maximum Mutual Information, the Kullback-Leibler Information (Cross Entropy), Penalized Maximum Likelihood Estimation and Smoothing Splines [6]. Averaging was shown to improve performance on the NIST OCR data, a human face recognition task and a time series prediction task [5].
J. Friedman (Stanford, "A New Approach to Multiple Outputs Using Stacking") presented a detailed analysis of a method for averaging estimators and noted simulations showed that averaging with a positivity constraint was better than cross-validation estimator selection [1].
S. Nowlan (Synaptics, "Competing Experts") emphasized the distinctions between static and dynamic algorithms and between averaged and stacked algorithms, and presented results of the mixture of experts algorithm [3] on a vowel recognition task and a hand tracking task.
H. Drucker (AT&T, "Boosting Compared to Other Ensemble Methods") reviewed the boosting algorithm [2] and showed how it can improve performance for OCR data.
J. Moody (OGI, "Predicting the U.S. Index of Industrial Production") showed that neural networks make better predictions for the US IP index than standard models [4] and that averaging these estimates improves prediction performance further.
W. Buntine (NASA Ames Research Center, "Averaging and Probabilistic Networks: Automating the Process") discussed placing combination techniques within the Bayesian framework.
D. Wolpert (Santa Fe Institute, "Inferring a Function vs. Inferring an Inference Algorithm") argued that theory can not, in general, identify the optimal network, so one must make assumptions in order to improve performance.
H. Thodberg (Danish Meat Research Institute, "Error Bars on Predictions from Deviations among Committee Members (within Bayesian Backprop)") raised the provocative (and contentious) point that Bayesian arguments support averaging while Occam's Razor (seemingly?) does not.
S. Hashem (Purdue University, "Merits of Combining Neural Networks: Potential Benefits and Risks") emphasized the importance of dealing with collinearity when using averaging methods.

*To appear in "Neural Information Processing Systems 6," S. J. Hanson, J. D. Cowan and C. L. Giles (eds.), Morgan Kaufmann, 1994.

References

[1] Leo Breiman. Stacked regression. Technical Report TR-367, Department of Statistics, University of California, Berkeley, August 1992.

[2] Harris Drucker, Robert Schapire, and Patrice Simard. Boosting performance in neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, [To appear].

[3] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(2), 1991.

[4] U. Levin, T. Leen, and J. Moody. Fast pruning using principal components. In Steven J. Hanson, Jack D. Cowan, and C. Lee Giles, editors, *Advances in Neural Information Processing Systems 6*. Morgan Kaufmann, 1994.

[5] M. P. Perrone. *Improving Regression Estimation: Averaging Methods for Variance Reduction with Extensions to General Convex Measure Optimization*. PhD thesis, Brown University, Institute for Brain and Neural Systems; Dr. Leon N Cooper, Thesis Supervisor, May 1993.

[6] M. P. Perrone. General averaging results for convex optimization. In *Proceedings of the 1993 Connectionist Models Summer School*, pages 364–371, Hillsdale, NJ, 1994. Erlbaum Associates.

[7] M. P. Perrone and L. N Cooper. When networks disagree: Ensemble method for neural networks. In *Artificial Neural Networks for Speech and Vision*. Chapman-Hall, 1993. Chapter 10.

Pulling It All Together: Methods for Combining Neural Networks
NIPS*93 Postconference Workshop
December 4, 1993

Intended Audience

Those interested in optimization algorithms for improving neural network performance.

Goal

Stimulate discussion on methods for combining neural networks

Organizer

Michael P. Perrone
Institute For Brain and Neural Systems, Brown University
Prometheus Inc., Newport RI
Email: mpp@cns.brown.edu

Pulling It All Together – NIPS'93 Workshop

7:30-7:35	Opening Remarks
7:35-7:55	M. Perrone, (Brown University) "Averaging Methods: Theoretical Issues and Real World Examples"
7:55-8:15	J. Friedman, (Stanford) "A New Approach to Multiple Outputs Using Stacking"
8:15-8:35	S. Nowlan, (Synaptics) "Competing Experts"
8:35-8:55	H. Drucker, (AT&T) "Boosting, Compared to Other Ensemble Methods"
8:55-9:30+	Discussion
9:30-4:30	FREE TIME
Cancelled	C. Scofield, (Nestor Inc) (Applications)
4:30-4:50	J. Moody (OGI) "Forecasting the U.S. Index of Industrial Production"
4:50-5:10	W. Buntine, (NASA Ames Research Center) "Averaging and Probabilistic Networks: Automating the Process"
5:10-5:30	D. Wolpert, (Santa Fe Institute) "Inferring a Function vs. Inferring an Inference Algorithm"
5:30-5:50	H. Thodberg, (Danish Meat Research Institute) "Error Bars on Predictions from Deviations among Committee Members (within Bayesian Backprop)"
5:50-6:10	S. Hashem, (Purdue University) "Merits of Combining Neural Networks: Potential Benefits and Risks"
6:10-6:30+	Discussion & Closing Remarks
7:00	Workshop Wrap-Up (common to all sessions)

Topics

- Boosting Algorithm
- Competing Experts Algorithm
- Ensemble Averaging Algorithm
- Stacked Generalization Algorithm
- Stacked Regression Algorithm
- Network Selection Techniques
- Bayesian Methods
- Cross Validation
- Incorporating A Priori Knowledge
- Error Orthogonality
- Task Decomposition
- Overcoming Data Sparsity
- Using Local Optima
- Overfitting

Averaging Methods: Theoretical Issues and Real World Examples

Michael P. Perrone

Institute for Brain and Neural Systems
Brown University
Email: mpp@cns.brown.edu

This research was supported by the Office of Naval Research, the Army Research Office and the National Science Foundation.

OUTLINE

- Problem of Multiple Neural Networks
- Ensemble Method
  - Basic Algorithm
  - Generalized Algorithm
  - Theoretical Basis
- Experimental Results
  - NIST OCR Database
  - Turk & Pentland Human Face Database
  - Sunspot Database
- Extensions
  - Convexity
  - Other Cost Functions
  - Variance Reduction

PROBLEM

$$\left.\begin{array}{l}\bullet \text{ Many learning algorithms}\\ \bullet \text{ Many possible architectures}\\ \bullet \text{ Many local minima}\end{array}\right\} \Rightarrow \begin{array}{c}\text{Many}\\ \textbf{disagreeing}\\ \text{networks}\end{array}$$

Naive estimate:

- Choose the best on an independent test set
- **We can do better!**

Combine networks:

- Bayesian BP (Buntine & Weigend 92)
- Hierarchical NNs (Ersoy & Hong 90)
- Hybrid NNs (Cooper 91; Scofield, et al 87; Reilly 88, 87)
- Local Experts (Jacobs, at al 91)
- Neural Trees (Perrone 92a, 92b; Sankar 90)
- Stacked Generalization (Wolpert 90)
- Synergy (Lincoln & Skrzypek 90)
- (Etc.)

⇒ Ensemble Method

- Simple
- Improves estimate
- Theoretical basis

Basic Ensemble Method

Given:

$(x, y) \sim \mathcal{P}$
$\mathcal{D} \equiv$ Training set
$\mathcal{CV} \equiv$ Cross-Validation set
$\mathcal{T} \equiv$ Testing set
Network set: $\{f_i(x)\}_{i=1}^{i=N}$ trained on $\mathcal{D}$

Find:

$$f(x) = E_{\mathcal{P}}[y|x]$$

Ensemble Regression Function:

$$f_{\text{ensemble}}(x) \equiv \frac{1}{N}\sum_{i=1}^{N} f_i(x)$$

Misfit:

$$m_i(x) \equiv f(x) - f_i(x)$$

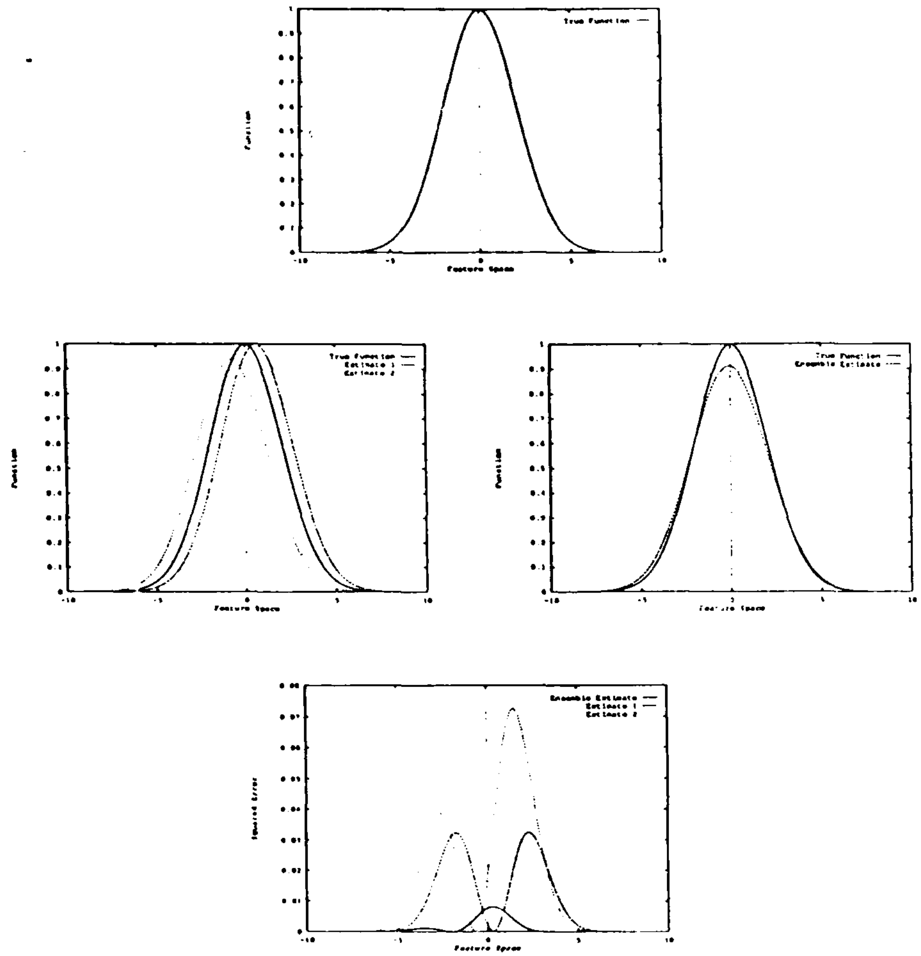
Average Individual MSE:

$$\overline{\text{MSE}}_{\text{individual}} = \frac{1}{N}\sum_{i=1}^{N} E_{\mathcal{CV}}[m_i^2]$$

Uncorrelated, Zero Mean $m_i(x)'s$:

$$\text{MSE}_{\text{Ensemble}} = \frac{1}{N}\overline{\text{MSE}}_{\text{Individual}}$$

INTUITIVE EXAMPLE



Removing the Independence Assumption

Cauchy Inequality

$$\Big(\sum_{i=1}^{n} x_i y_i\Big)^2 \leq \Big(\sum_{i=1}^{n} x_i^2\Big)\Big(\sum_{i=1}^{n} y_i^2\Big)$$

$$\Big(\sum_{i=1}^{n} x_i\Big)^2 \leq n \sum_{i=1}^{n} x_i^2$$

$$\mathrm{MSE}[\bar{f}] \leq \overline{\mathrm{MSE}[f]}$$

$$\sum_{i=1}^{n} (y_i - f(x_i))^2 / g^2(x_i)$$

Connection to Monte Carlo Methods

$$y = \int f(x;w)P(w)dw.$$

$$E_w[f(x;w)] \approx \frac{1}{n}\sum_{i=1}^{n} f(x;w_i)$$

Smoothing by Variance Reduction

$$\begin{aligned}
E[(\hat{f}(x) - f(x))^2] &= E[(\hat{f} - E[\hat{f}] + E[\hat{f}] - f(x))^2] \\
&= E[(\hat{f} - E[\hat{f}])^2] + E[(E[\hat{f}] - f(x))^2] \\
&\quad +2E[(\hat{f} - E[\hat{f}])(E[\hat{f}] - f(x))] \\
&= \mathrm{VAR}(\hat{f}) + \mathrm{BIAS}^2(\hat{f})
\end{aligned}$$

$$\mathrm{VAR}(\hat{f}) \equiv E[(\hat{f} - E[\hat{f}])^2]$$

$$\mathrm{BIAS}(\hat{f}) \equiv E[\hat{f}] - f(x)$$

$$\lim_{n\to\infty} \overline{f}(x) = \int \hat{f}(x;\beta)p(\beta)d\beta$$

Generalized Ensemble Method

Weighted Average:

$$\hat{f}(x) \equiv \sum_{i=1}^{i=N} \alpha_i f_i(x), \qquad \alpha_i \in \mathcal{R}$$

$$\sum \alpha_i = 1 \quad \Rightarrow \quad \hat{f}(x) = f(x) + \sum_{i=1}^{i=N} \alpha_i m_i(x)$$

Covariance Matrix:

$$C_{ij} \equiv E_{CV}[m_i(x) m_j(x)]$$

Minimize:

$$\mathrm{MSE}(\alpha) = \sum_{i,j} \alpha_i \alpha_j C_{ij}$$

Result:

$$\alpha_i^{\mathrm{opt}} = \frac{\Sigma_j C_{ij}^{-1}}{\Sigma_k \Sigma_j C_{kj}^{-1}} \quad \text{and} \quad \mathrm{MSE}(\alpha^{\mathrm{opt}}) = \left[\sum_{ij} C_{ij}^{-1}\right]^{-1}$$

Uncorrelated $m_i(x)'s$:

$$\alpha_i^{\mathrm{opt}} = \frac{\sigma_i^{-2}}{\Sigma_j \sigma_j^{-2}} \quad \text{and} \quad \mathrm{MSE}(\alpha^{\mathrm{opt}}) = \left[\sum_i \sigma_i^{-2}\right]^{-1}$$

Unconstrained Ensemble Method

Weighted Average:

$$\hat{f}(x) \equiv \sum_{i=1}^{i=N} \alpha_i f_i(x), \qquad \alpha_i \in \mathcal{R}$$

Estimate Matrix and Measurement Vector:

$$f_{ji} \equiv f_i(x_j)$$
$$F_j \equiv f(x_j)$$

Minimize:

$$\mathrm{MSE}(\alpha) = (f\alpha - F)^{\mathrm{T}}(f\alpha - F)$$

Result:

$$\alpha = (f^{\mathrm{T}} f)^{-1} f^{\mathrm{T}} F$$

Infinite Data:

$$(f^{\mathrm{T}} f)_{ij} \rightarrow E\,[f_i(x) f_j(x)]$$
$$(f^{\mathrm{T}} F)_i \rightarrow E\,[f_i(x) f(x)]$$

Convexity and Averaging

Convex on $[a, b]$

$$h\left(\frac{x_1 + x_2}{2}\right) \leq \frac{h(x_1) + h(x_2)}{2}$$

Jensen's inequality

$$\Phi\left(\frac{\int_a^b f(x;\omega)g(\omega)d\omega}{\int_a^b g(\omega)d\omega}\right) \leq \frac{\int_a^b \Phi(f(x;\omega))g(\omega)d\omega}{\int_a^b g(\omega)d\omega}$$

$$\Phi(Ef) \leq E[\Phi(f)]$$

$$\Phi(\bar{f}) \leq \overline{\Phi(f)}$$

l_p-norm Extension

$$E(\{x_j\}) = \sum_{ij}(\alpha_i |x_j|^{p_{\alpha i}} - \beta_i |x_j|^{p_{\beta i}})$$

Extensions to Other Cost Functions

Entropy $\quad H(p) = -\sum_i p(x_i) \ln p(x_i)$

$$H(\bar{p}) \geq \overline{H(p)}$$

Mutual Information

$$I(a, b) = H(a) + H(b) - H(ab)$$

$$I(\bar{p}, b) \geq \overline{I(p, b)}$$

Log-Likelihood Function

$$L(p) = \prod_i p(x_i)$$

$$\ln L(\bar{p}) \geq \overline{\ln L(p)}$$

Kullback-Leibler Information

$$K(f, g) = \int f \ln\left(\frac{f}{g}\right)$$

$$\overline{K(p, g)} \geq K(\bar{p}, g)$$

Smoothing Splines

$$S(f) = \frac{1}{n}\sum_i(\hat{f}(x_i) - f_i)^2 + \lambda \int (f'')^2 dx$$

$$S(\bar{f}) \leq \overline{S(f)}$$

Experimental Results

- BP nets with 2 weight layers
- Cross-validatory stopping rule
- Varied the number of hidden units
- Random initial weights
- 10 nets in each ensemble
- Figure of Merit:

$$\text{FOM} \equiv \%\text{Correct} - \%\text{Rejected} - 10(\%\text{Error})$$

NIST OCR Data

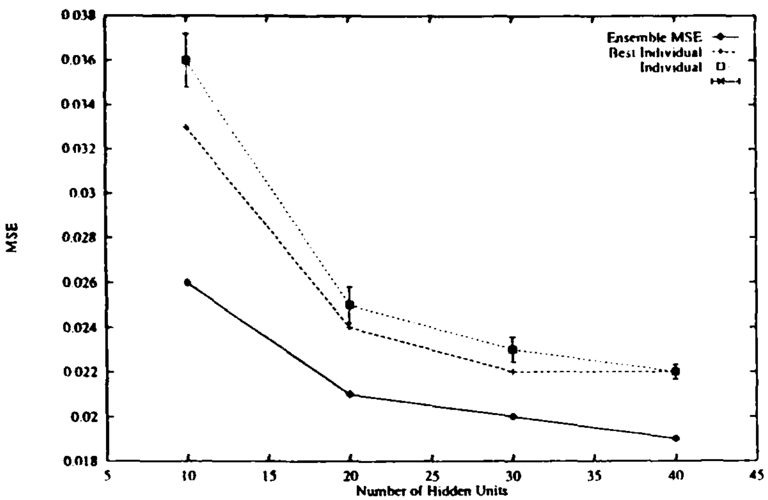
- Handwritten characters
- Hand-segmented
- Hand-labeled
- 120 dimensional feature space

(Supplied by Nestor, Inc., Providence, RI)

DATA SET	TRAINING SET	CV SET	TESTING SET	CLASSES
Numbers	13241	13241	4767	10
Uppercase	11912	11912	7078	26
Lowercase	12971	12970	6835	26

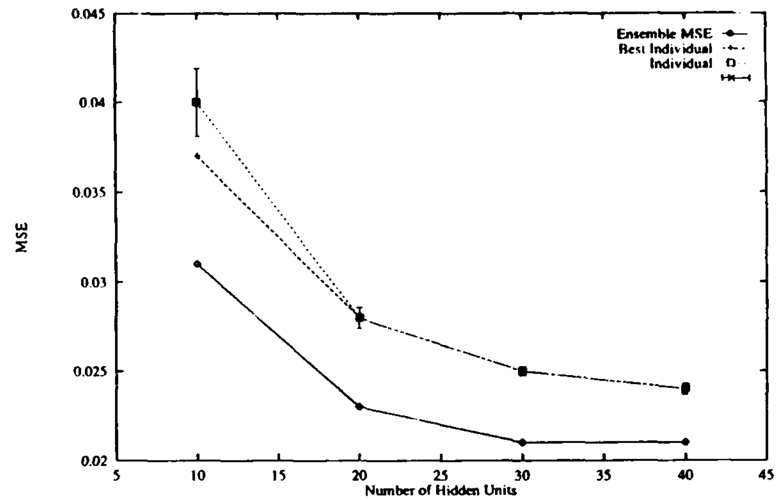
MSE vs. Hidden Units

NIST Uppercase - Independent test set



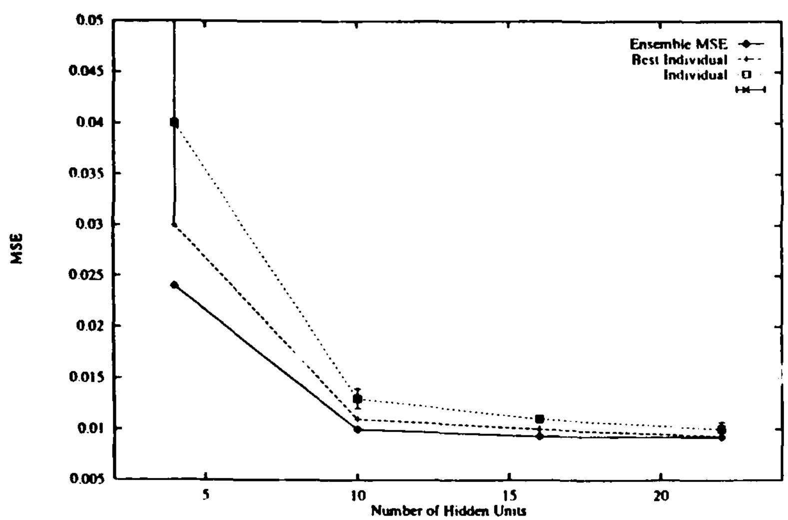
MSE vs. Hidden Units

NIST Lowercase - Independent test set



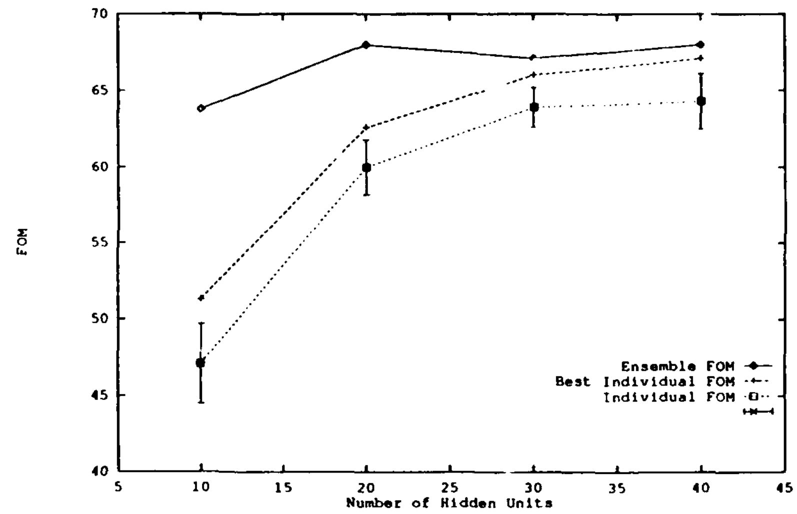
MSE vs. Hidden Units

NIST Numerals - Independent test set

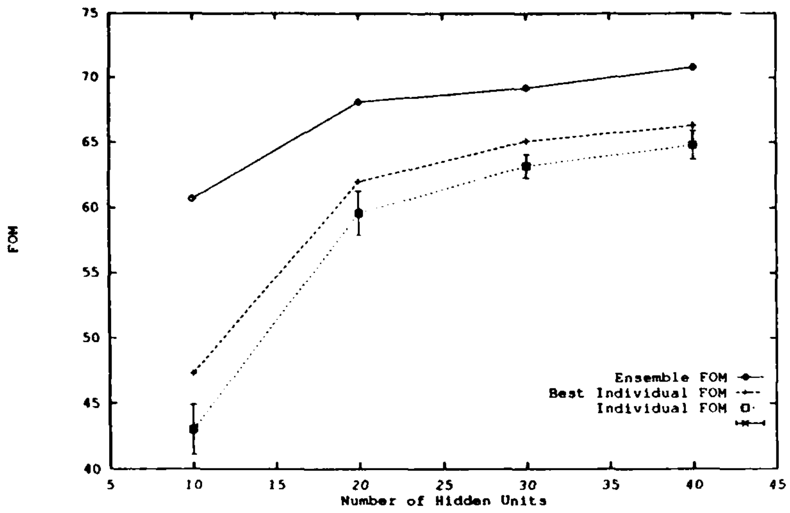


FOM vs Hidden Units

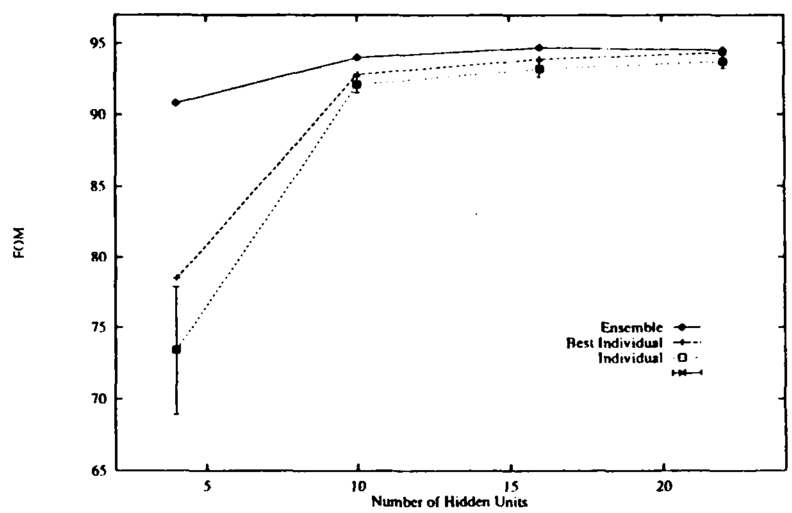
NIST Uppercase - Independent test set



FOM vs Hidden Units
NIST Lowercase - Independent test set



FOM vs Hidden Units
NIST Numbers - Independent test set



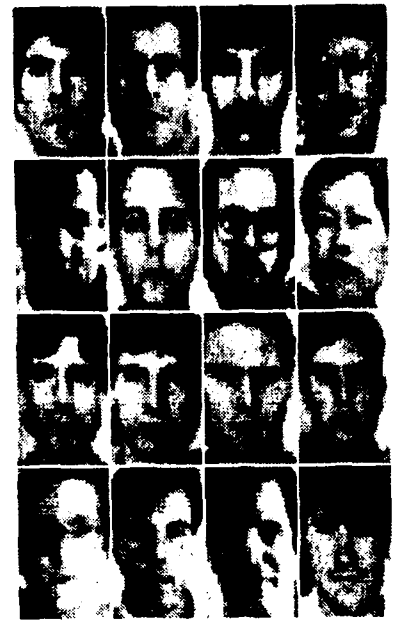
Turk and Pentland Data

- Human male face images
- "Warped" - centered and rotated
- 2294 dimensional gray-scale feature space

(Supplied by Daniel Reisfeld, Tel Aviv University)

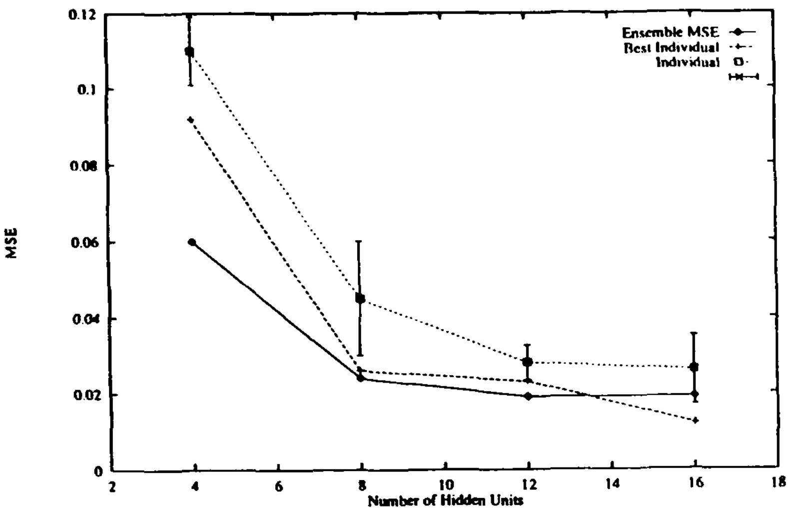
DATA SET	TRAINING SET	CV SET	TESTING SET	CLASSES
Faces	136	136	160	16

Samples of the Face Data



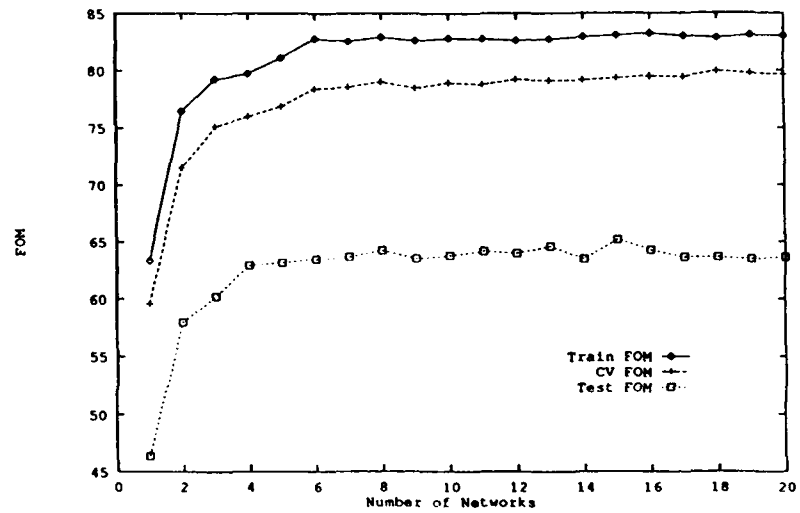
MSE vs. Hidden Units

Human Face Data - Independent test set



DEPENDENCE ON ENSEMBLE SIZE

FOM vs Number of Ensemble Nets
NIST Uppercase - 10 hidden units



Network Selection

- Multiple Data Sets
- Multiple Architectures
- Multiple Learning Rules - flexibility!
- Bootstrap Generation
  - One training set
  - Multiple trained nets
- Generalized Jackknife Generation
  - Vary the CV subset
  - Uses entire data set AND avoids over-fitting

SUMMARY

PROBLEM: Neural nets often disagree

GENERALIZED ENSEMBLE METHOD:

- Construct ensemble: **weighted average** of networks
- Choose weights by minimizing MSE (with nets fixed)

ADVANTAGES:

- Simple
- Firm theoretical basis
- Significantly improves performance
- General - Combines multiple architectures data sets and learning rules
- Extends to any convex cost optimization

A New Approach to Multiple Outputs Through Stacking

Leo Breiman
U.C. Berkeley

Jerome H. Friedman
Stanford University

<u>Problem</u>:

<u>model</u>: $\{ \underset{\uparrow \text{ outputs}}{y_m} = \underset{\text{target functions}}{f_m} \underbrace{(x_1 \cdots x_m)}_{\text{common inputs}} + \underset{\substack{\uparrow \\ \text{noises} \\ (\text{maybe} = 0)}}{\varepsilon_m} \}_1^M$

<u>training data</u>: $\{ y_{i1} \cdots y_{im} \,;\, x_{i1} \cdots x_{im} \}_1^N$

<u>develop</u>: $\{ \hat{f}_m (x_1 \cdots x_m) \}_1^M$ approximations

<u>such that</u>:

$$\{ E[f_m(\underline{x}) - \hat{f}_m(\underline{x})]^2 = \underset{(\text{each})}{\text{small}} \}_1^M$$

<u>def</u>: $\underline{x} = \{ x_1 \cdots x_n \} \in R^n$

<u>Approach (1)</u>: separate problems

$$\hat{f}_m(\underline{x}) = \underset{g(\underline{x})}{\text{argmin}} \underbrace{\sum_{i=1}^{N} [y_{im} - g(\underline{x}_i)]^2}_{\text{fit to the training data}} + \lambda_m P_m(g)$$

regularization - $\begin{cases} \text{strength parameter} \\ \text{penalty functional} \end{cases}$

$$\hat{f}_m(\underline{x}) = \sum_{k=1}^{K_m} \hat{a}_{mk} \underset{\substack{\uparrow \\ \text{basis functions}}}{B} (\underline{x} \mid \underset{\substack{\uparrow \\ \text{parameters}}}{\hat{\underline{\gamma}}_{mk}})$$

<u>Approach (2)</u>: common estimation <u>(usual)</u>

$$\{ \hat{f}_m(\underline{x}) \}_1^M = \underset{\{g_m(\underline{x})\}_1^M}{\text{argmin}} \sum_{m=1}^{M} \sum_{i=1}^{N} [y_{im} - g_m(\underline{x}_i)]^2$$

$$+ \underset{\substack{\uparrow \\ \text{single strength parameter}}}{\lambda} \underset{\substack{\uparrow \\ \text{common penalty}}}{P} (g_1 \cdots g_M)$$

$$\hat{f}_m(\underline{x}) = \sum_{k=1}^{K} \hat{a}_{mk} \underset{\substack{\uparrow \\ \text{common basis function set}}}{B} (\underline{x} \mid \hat{\underline{\gamma}}_k)$$

FFNN's, PPR, CART, MARS, RBF's etc

Motivations for (2) over (1):

A. Comptational

B. Statistical

if $\{f_m(\underline{x})\}_1^M$ all quite similar

then they can borrow strength from each other => fewer (nonlinear) parameters.

Additional strength borrowing:

let $\{z_k = B(\underline{x} \mid \underline{\gamma}_k)\}_1^K$ "transformed inputs"

$$\hat{f}_m(\underline{x}) = \sum_{k=1}^{K} \hat{a}_{mk} z_k \quad \text{linear model}$$

$$\{\hat{a}_{mk}\} = \underset{\{a_{mk}\}}{\operatorname{argmin}} \sum_{m=1}^{M} \sum_{i=1}^{N} \left[y_{im} - \sum_{k=1}^{K} a_{mk} z_{ik}\right]^2$$

linear least squares regression

Breiman & Friedman (1993): "C&W"

Instead of $\{\hat{f}_m(\underline{x})\}_1^M$, use $\{\tilde{f}_m(\underline{x})\}_1^M$

$$\tilde{f}_m(\underline{x}) = \sum_{l=1}^{M} b_{ml} \hat{f}_l(\underline{x}) \quad \text{linear combinations}$$

let: $\hat{\underline{f}}(\underline{x}) = \{\hat{f}_m(\underline{x})\}_1^M \in R^M$

$\tilde{\underline{f}}(\underline{x}) = \{\tilde{f}_m(\underline{x})\}_1^M \in R^M$

$B = [b_{ml}] \in R^{M \times M}$

then: $\tilde{\underline{f}}(\underline{x}) = B\, \hat{\underline{f}}(\underline{x})$

least squares estimate

"shrinking" matrix

How to get B?

ideally:

$$\{b_{ml}\}_1^M = \underset{\{b_l\}_1^M}{\operatorname{argmin}}\, E\left[y_m - \sum_{l=1}^{M} b_l \hat{f}_l(\underline{x})\right]^2$$

future data (population)

optimizes each output error separately

Don't have future data

$\Rightarrow$ cross-validation

- $\{\hat{b}_{m\ell}\}_1^M = \underset{\{b_\ell\}_1^M}{\arg\min} \sum_{i=1}^N [y_{im} - \sum_{\ell=1}^M b_\ell \hat{f}_{\ell\backslash i}(\underline{x}_i)]^2$

- $\hat{f}_{\ell\backslash i} = \hat{f}_\ell(\underline{x})$ estimated with ith training datum not in training sample.

uses $\{\hat{f}_\ell(\underline{x})\}_1^M$ as "stacked" approx's for each $\hat{f}_m(\underline{x})$, $m = 1, M$.

Breiman & Friedman (1993):

$\ell e = [\hat{F}_{i\ell}] = [\hat{f}_\ell(\underline{x}_i)]$, $Y = [y_{i\ell}]$, $\in R^{N \times M}$

$$\varphi = (Y^T Y)^{-1} Y^T \hat{F} (\hat{F}^T \hat{F})^{-1} \hat{F}^T Y \in R^{M \times M}$$

$$= T D_g T^{-1} \quad \text{(eigen analysis)}$$

$T \in R^{M \times M}$ = eigenvectors of φ (canonical coordinat

$D_g = \text{diag}\{g_1^2 \cdots g_M^2\}$ = eigen values

(canonical correlations squared

then: $\hat{B} \simeq T D_b T^{-1}$ (GCV approx. to CV

$D_b = \text{diag}\{b_1 \cdots b_M\}$

$$b_i = \frac{(1 - K/N)(g_i^2 - K/N)}{(1-K/N)^2 g_i^2 + (K/N)^2 (1 - g_i^2)}$$

("magic formula")

Breiman & Friedman show

$$b_i \leftarrow [b_i]_+ = \begin{cases} b_i & \text{if } b_i \geq 0 \\ 0 & \text{if } b_i < 0 \end{cases}$$

(positive part shrinkage)

is a BIG HELP.

(works better than real cross-validation.

<u>Good news</u>: C&W "post processor" can help with approach (2) to reduce prediction error for <u>each</u> output.

<u>Bad news</u>: approach (2) does <u>not</u> have this property - why?

(1) <u>Wrong criterion</u>:

let $\underline{y} = \{y_m\}_1^M$, $\underline{f}(\underline{x}) = \{f_m(\underline{x})\}_1^M \in R^M$

$$\Sigma(\underline{f}) = E[\underline{y} - \underline{f}(\underline{x})][\underline{y} - \underline{f}(\underline{x})]^T$$

residual (error) covariance matrix (population - future data)

<u>Correct criterion</u>:

$$\hat{\underline{f}}(\underline{x}) = \underset{\underline{g}(\underline{x})}{\operatorname{argmin}} \sum_{i=1}^{N} [\underline{y}_i - \underline{g}(\underline{x}_i)]^T \Sigma^{-1}(\underline{f}) [\underline{y}_i - \underline{g}(\underline{x}_i)] + \lambda P(\underline{g})$$

(usual) wrong criterion $\Rightarrow \Sigma(\underline{f}) = I_M$

$\underline{f}(\underline{x})$ unknown $\Rightarrow \Sigma(\underline{f})$ unknown

$$\hat{\Sigma}(\hat{\underline{f}}) = \frac{1}{N} \sum_{i=1}^{N} [\underline{y}_i - \hat{\underline{f}}(\underline{x}_i)][\underline{y}_i - \hat{\underline{f}}(\underline{x}_i)]^T$$

<u>and iterate</u>.

this is equivalent to:

$$\hat{\underline{f}}(\underline{x}) = \underset{\underline{g}(\underline{x})}{\operatorname{argmin}} \log |\hat{\Sigma}(\underline{g})| + \lambda P(\underline{g})$$

setting $\Sigma = I_M$ gives too much influence to poorly estimated $f_m(\underline{x})$ <u>(disaster)</u>

(2) <u>common penalty</u> $\Rightarrow$ common basis function set.

if $\{f_m(\underline{x})\}_1^M$ not very similar,

then compromise $\{B_k(\underline{x} \mid \underline{\gamma}_k)\}_1^K$ can be bad for some (or all) $\{\hat{f}_m(\underline{x})\}_1^M$

Especially if $\Sigma = I \Rightarrow$ poorly estimatable $\hat{f}_m(\underline{x})$ dominate basis function selection.

(3) Single regularization (strength) parameter:

- compromise value
- $\hat{\lambda} = \underset{\lambda}{\operatorname{argmin}} \sum_{m=1}^{M} \widehat{mse}[\hat{f}_m(\underline{x} \mid \lambda)]$

estimate of future prediction error

can be bad for some (or all) $\hat{f}_m(\underline{x})$

Poorly estimated $\hat{f}_m(\underline{x})$ dominate here too.

∴ Approach (2) with / without C&W can make things worse (much) than separate approx's (approach (1)),

especially for $f_m(\underline{x})$ that can be approx'ed well.

<u>Desire</u>: borrow strength in a way that <u>insures</u> (expected) improvement over separate approx's (approach (1)).

(A) If $\{f_m(\underline{x})\}_1^M$ are very different ⇒ ~ same as separate approx's.

(B) If they are highly correlated ⇒ big improvement.

How?

<u>Idea (1)</u>: apply C&W to <u>separate</u> approx's.

$$\hat{f}_m(\underline{x}) = \underset{g(\underline{x})}{\operatorname{argmin}} \sum_{i=1}^{N} [y_{im} - g(\underline{x}_i)]^2 + \lambda_m P_m(g)$$

$$\Phi = (Y^T Y)^{-1} Y^T \hat{F} (\hat{F}^T \hat{F})^{-1} \hat{F}^T Y = T D_g T^{-1}$$

$$\hat{B} = T D_b T^{-1}; \quad D_b = \text{magic formula } \{g_i^2\}_1^M$$

$$K = \text{ave \# of parameters} = \frac{1}{M} \sum_{m=1}^{M} p_m(\text{eff})$$

$$\tilde{f}_m(\underline{x}) = \sum_{l=1}^{M} \hat{b}_{ml} \hat{f}_m(\underline{x}), \quad m = 1, M$$

final approx's.

Our implementation: always include $\hat{f}_m(\underline{x})$'s original basis functions in $\hat{f}^*_m(\underline{x})$:

$$\hat{f}^*_m(\underline{x}) = \underbrace{\sum_{k=1}^{K_m} \hat{a}^*_{mk} B(\underline{x} \mid \hat{\underline{\gamma}}_{mk})}_{\text{original basis fun.'s}} + \underbrace{\sum_{l=1}^{L_m} \hat{a}^*_{ml} B(\underline{x} \mid \hat{\underline{\gamma}}_{lr})}_{\text{added basis functions from other } \hat{f}_m(x) \text{ (if any)}}$$

If any $\{f_l(\underline{x})\}_{l \neq m}$ highly associated with $\hat{f}_m(\underline{x})$ => their basis functions may help $f_m(\underline{x})$. <u>Note</u>: Some basis fun's will appear in several $\hat{f}^*_m(\underline{x})$.

<u>Then</u>: Apply C&W to $\{\hat{f}^*_m(\underline{x})\}_1^M$

(Idea 2 + Idea 1)

$$\tilde{f}^*_m(\underline{x}) = \sum_{l=1}^{M} \hat{b}_{ml} \hat{f}^*_m(\underline{x})$$

<u>Idea (2)</u>: Share basis functions in a way that does not degrade (separate) performance.

$$\{\hat{f}_m(\underline{x}) = \sum_{k=1}^{K_m} \hat{a}_{mk} B(\underline{x} \mid \underbrace{\hat{\underline{\gamma}}_{mk}}_{\text{fit to } \{y_{im}\}_1^N \text{ only}})\}_1^M \quad \underline{\text{separate}}$$

merge separate basis functions of all outputs into a common pool

$$\{c_l(\underline{x})\}_1^K = \{B(\underline{x} \mid \hat{\underline{\gamma}}_{mk})\}_{k=1}^{K_m}{}_{m=1}^{M}$$

$$K = \sum_{m=1}^{M} K_m$$

and select individual (optimal) subsets for each $\hat{f}_m(\underline{x})$ => separate model selection for each $\hat{f}_m(\underline{x})$.

$$\hat{f}^*_m(\underline{x}) = \sum_{\hat{a}^*_{ml} \neq 0} \hat{a}^*_{ml}\, c_l(\underline{x}).$$

Simulation studies: 100 replications per study.

each replication (data set):

$\{ f_m(\underline{x}) = \sum_{\ell=1}^{M} a_{\pi(\ell)}\, b_{m\ell}\, B_\ell(\underline{x}) \}_1^M$

$\{ B_\ell(\underline{x}) \}_1^M$ = fixed prespecified fun's (uncorrelated)

$b_{\ell m} \sim N(0, 1)$ random

π = random permutation $[1 \dots M]$

$a_\ell = \exp[-0.5 (\ell / \bar{\ell})^2]$

$\bar{\ell}$ controls correlation among $\{f_m(\underline{x})\}_1^M$

$\{ y_{im} = f_m(\underline{x}_i) + \sigma \cdot \varepsilon_i \}_{i=1}^{N}{}_{m=1}^{M}$

$\underline{x}_i \sim U^n(0, 1)$, $\varepsilon_i \sim N(0, 1)$

σ controls noise level.

$\{ f_m(\underline{x}) \}_1^M$ (randomly) different for each replication

corr's among $\{ f_m(\underline{x}) \}_1^M \sim$ const over the 100 rep's.

(100·M diff. target fun's / study)

For each replication:

$\hat{f}_m(\underline{x})$ = separate approx's in order of $E[f_m(\underline{x}) - \hat{f}_m(\underline{x})]^2$.

$\tilde{f}_m(\underline{x})$ = corresponding post-processed approx's.

$$r_m^2 = \frac{E[f_m(\underline{x}) - \tilde{f}_m(\underline{x})]^2}{E[f_m(\underline{x}) - \hat{f}_m(\underline{x})]^2}$$ "improvement ratio."

look at distribution $\{ \overline{r_m^2} \}_1^M$ averaged over 100 rep's.

$E[\cdot]$ = ave. over 5000 indep. validation sample.

Study 1 : $M = 7$, $n = 2$, $N = 200$

$B_1 = \sin(2\pi x_1)$, $B_2 = \sin(4\pi x_1)$

$B_3 = \sin(2\pi x_2)$, $B_4 = \sin(4\pi x_2)$

$B_5 = \sin(2\pi x_1) \sin(2\pi x_2)$

$B_6 = \sin(2\pi x_1) \sin(4\pi x_2)$

$B_7 = \sin(4\pi x_1) \sin(2\pi x_2)$

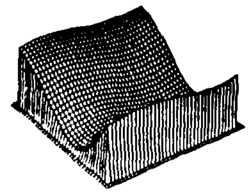
8 cases (100 rep's each):

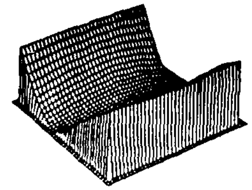
ave $|\mathrm{corr}\{f_m(\underline{x})\}_1^M| = 0.8, 0.66, 0.55$
0.33

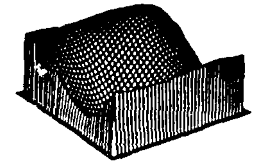
$\sigma = 0.0$, $\sigma = 0.2$

left = add basis fun's only

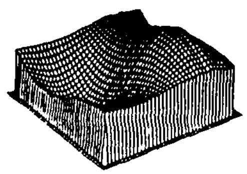
right = add BF's + C&W ("stacking")



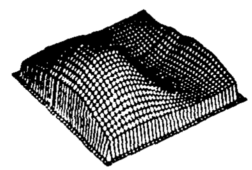




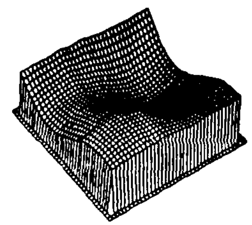
surf 4



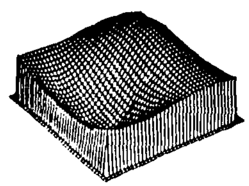
surf 5



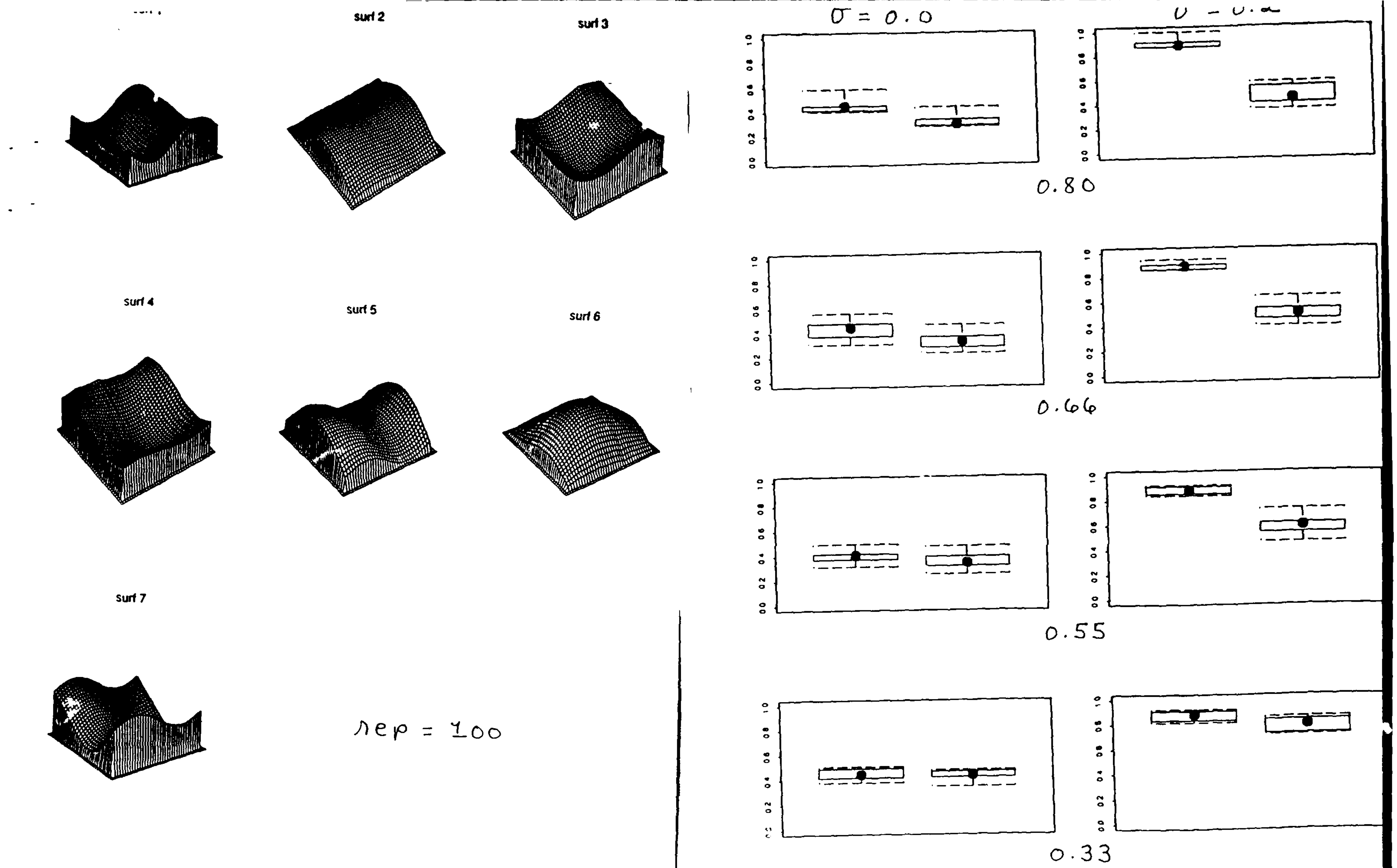
surf 6



surf 7



rep = 50


surf 2
surf 3
surf 4
surf 5
surf 6
surf 7
rep = 100
σ = 0.0
0.80
0.66
0.55
0.33
0.0
0.2
0.4
0.6
0.8
1.0

Study 2: $M = 9, \; n = 9, \; N = 200$

$\{B_\ell(\underline{x})\}_1^9 = \{\sin(5\pi x_\ell)\}_1^9$

independent

$\therefore \; \{ f_m(\underline{x}) = \sum_{\ell=1}^{M} a_{\pi(\ell)} \, b \quad B_\ell(\underline{x}) \}_1^M$

additive

Case 1: $b_{m\ell} \sim N(0, 1)$, $\bar{\ell} = 2$

ave $|\operatorname{corr} \{f_m(\underline{x})\}_1^M| = 0.65$

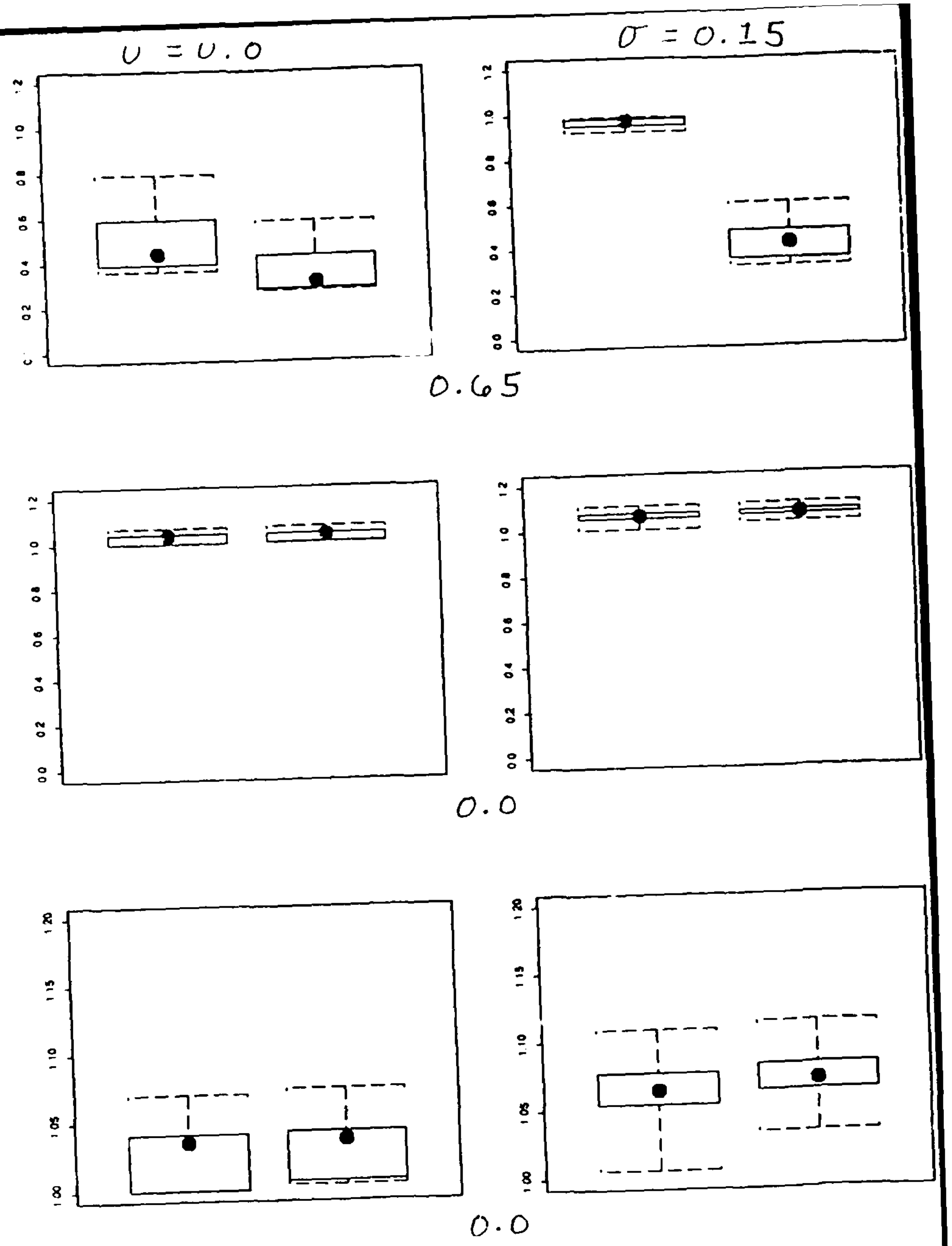
$\sigma = 0.0, \; 0.15$

Case 2: $b_{m\ell} = \delta_{m\ell} = \begin{cases} 1 & m = \ell \\ 0 & m \neq \ell \end{cases}$

$\Rightarrow \; \{f_m(\underline{x})\}_1^M$ = independent

$\Rightarrow$ separate $\{\hat{f}_m(\underline{x})\}_1^M$ = optimal

Question: How much worse are the post processed $\{\tilde{f}_m(\underline{x})\}_1^M$?



<u>Study 3</u>: $M = 9$, $n = 2$, $N = 200$

$\{B_\ell(\underline{x}) = e^{-\frac{1}{2}|\underline{x} - \underline{c}_\ell|^2 / s_\ell^2}\}_1^9$

$\underline{x} \sim U^2(0, 4)$, $s_\ell \sim U(0.7, 1.3)$

$\{\underline{c}_\ell\}_1^9 = (1,1), \ldots, (3,3)$

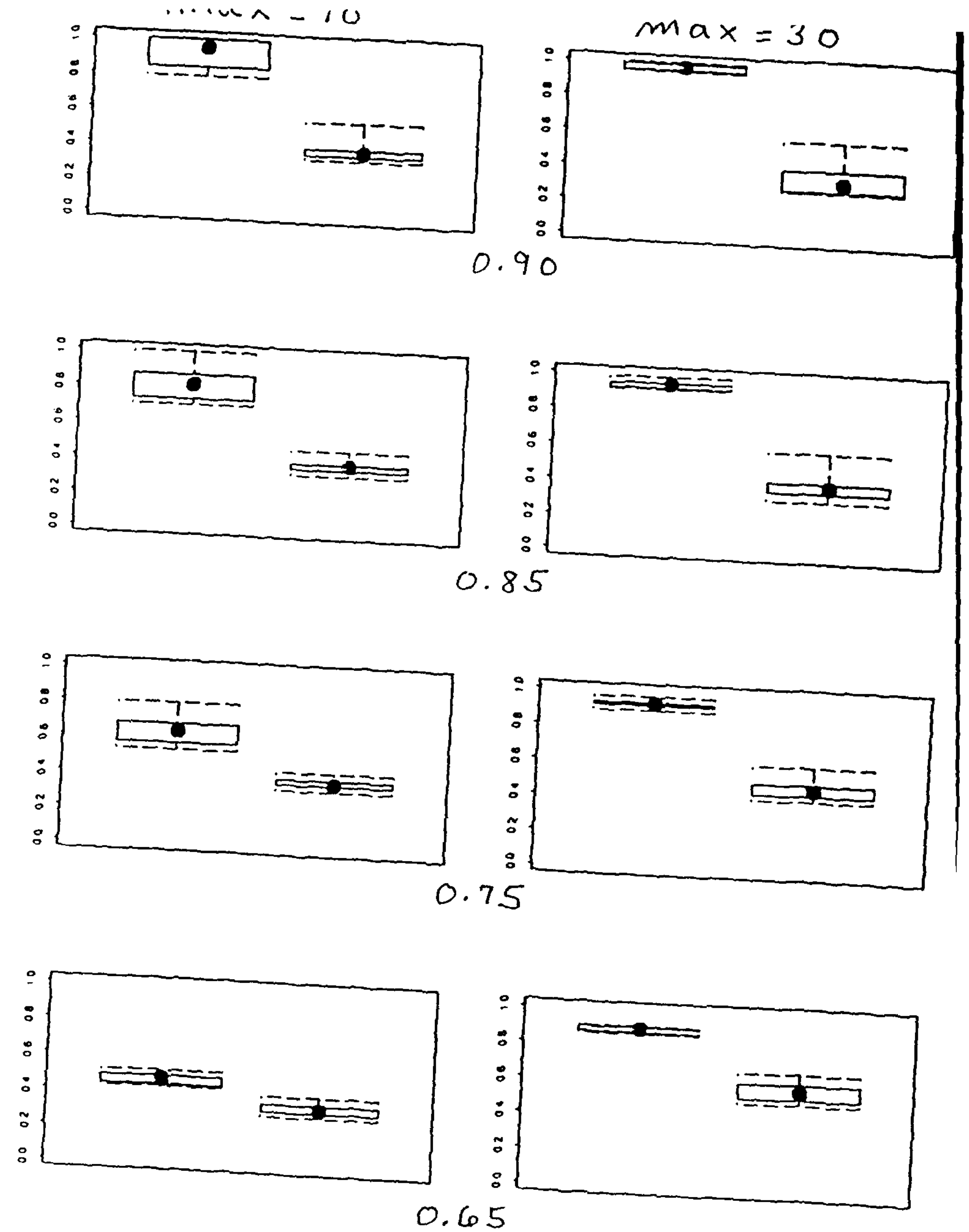
$\sigma = 0.08$

ave $|\text{corr}\{f_m(\underline{x})\}_1^M| =$

0.90 0.85 0.75 0.65

max. # of basis fun's (MARS) =

10 30



Mixtures of Experts Revisited

Steven J. Nowlan
Synaptics

Mixture of Experts

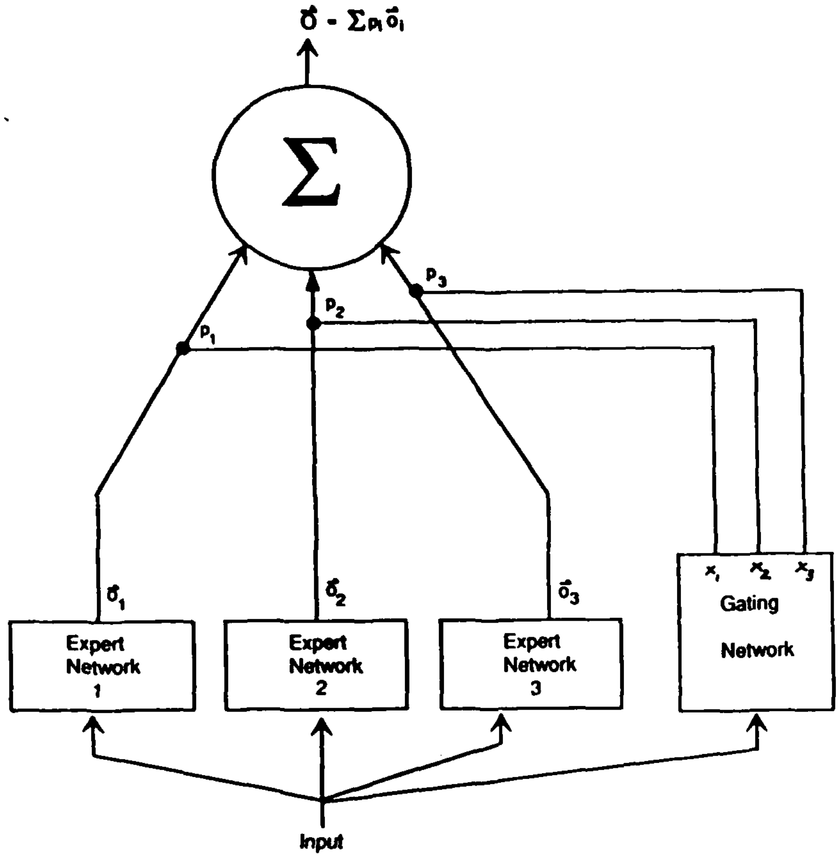
A modular architecture which decomposes a problem into a set of quasi-independent tasks:

- Each module is a local expert which learns to perform one or more tasks. Experts compete for the right to learn each training pattern.

- Gating network mediates the competition among experts. Like a manager it assigns different problems to different experts.

- Output is a **weighted average** of experts:

$$\vec{O} = \sum_i p_i \vec{o}_i$$



1. Objective: Max $\log L = \Sigma_c \, log P(\vec{d_c}|\vec{I_c})$
2. Error for expert i: $E_i = p(E_i|\vec{d_c})(\vec{d_c} - \vec{o}_i)$
3. Errors for gating net: $E_g(x_i) = p(E_i|\vec{d_c}) - p_i$

Mixture Equations

Objective:

$$\text{MAX } \log L = \sum_c \log P(\vec{d_c})$$

where

$$P(\vec{d_c}) = \sum_{i=1}^{N} p_i \, p(\vec{d_c}|i)$$

The probability of expert j generating the desired output is:

$$p(\vec{d_c}|j) = \frac{1}{K\sigma_j} e^{-\frac{\|\vec{d_c} - \vec{o}_j(\vec{I_c})\|^2}{2\sigma_j^2}}$$

The gating network outputs are normalized with a "soft max":

$$p_j(\vec{I_c}) = \frac{e^{x_j^p}}{\Sigma_i \, e^{x_i^p}}$$

Expert network cost derivatives:

$$\frac{\partial \log L}{\partial \vec{o}_j} = \frac{1}{\sigma_j^2} \sum_c p(j|\vec{d_c}) \, (\vec{d_c} - \vec{o}_j)$$

where

$$p(j|\vec{d_c}) = \frac{p_j \, p(\vec{d_c}|j)}{\Sigma_i \, p_i \, p(\vec{d_c}|i)}$$

Gating network cost derivatives:

$$\frac{\partial \log L}{\partial x_j^p} = \sum_c p(j|\vec{d_c}) - p_j$$

Approaches to Combining Networks

	Static	Dynamic
Averaging	Committees GEM Clustering Networks	Mixture of Experts Meta-Pi
Stacking	Stacked Generalizers Boosting	Constructive M.E.?

Mixture of Experts:

- Output = a convex combination of expert outputs.
- Weights → computed dynamically as a function of input to gating net.
- Gating net with only thresholds:

$$w_j = \langle p(E_j|d) \rangle$$

Maximizing Effectiveness

Like other averaging schemes, the mixture of experts can be more effective by attempting to minimize the correlation between expert errors.

Examples:

1. Gating network input is different from expert network input.

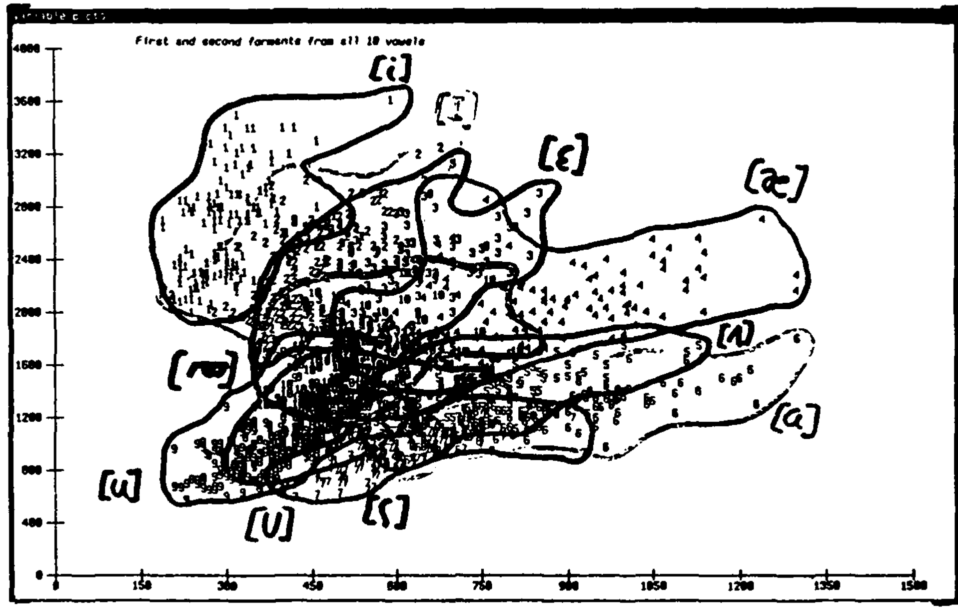
2. Expert network inputs are different (fusion).

Experiments on 10 Vowel Data Set

Data – First and second formants from 2 repetitions of 10 different vowels from 75 speakers (32 Male, 28 Female, 15 Children). From Peterson & Barney, 1952.

Systems Simulated

- A variety of multi-layer back propagation (BP) networks for benchmarking.
- Mixtures of experts where each expert received the formants as input, but the input to the gating network varied:
  - formant values
  - speaker identity
  - male, female or child (MFC)
  - MFC + min and max values of F1 and F2 for each speaker
  - min and max values of F1 and F2 for each speaker



Word List for Vowel Study

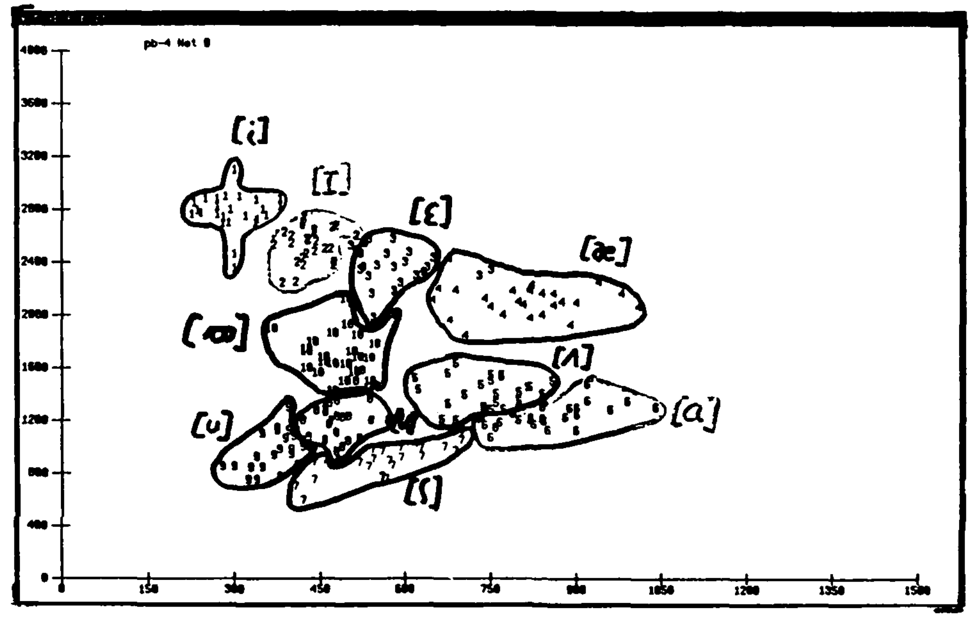
Word	Vowel Symbol	Class Number
heed	[i]	1
hid	[I]	2
head	[ε]	3
had	[æ]	4
hud	[Λ]	5
hod	[a]	6
hawed	[ς]	7
hood	[U]	8
who'd	[u]	9
heard	[ϖ]	10

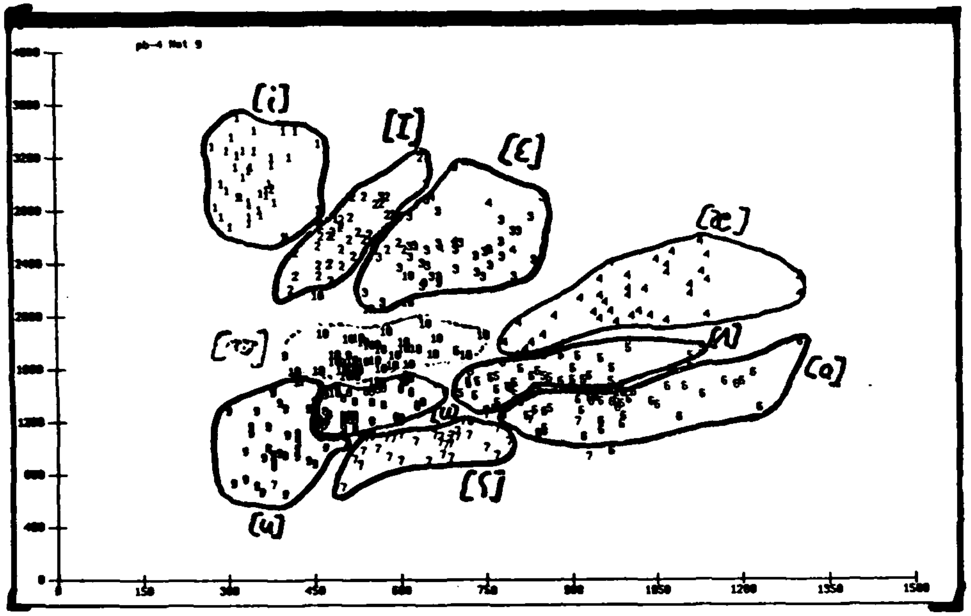
Comparison to Single BP Network
Generalization on Test Data

Type of Input	Mixture Error % (Test)	BP Error % (Test)	Sig.(p)
Formants only	15.1 ± 0.9	23.3 ± 1.2	$\gg 0.9999$
Form. + Speaker ID	6.4 ± 1.3	-	$\gg 0.9999$
Form. + MFC	13.5 ± 0.6	18.4 ± 1.1	$\gg 0.9999$
Form. + MFC + Range	6.2 ± 0.9	16.1 ± 1.0	$\gg 0.9999$
Form. + Range	12.8 ± 0.9	16.2 ± 0.8	> 0.9999

Example Speaker Decomposition

Expert #	% Male	% Female	% Child	% Total
0	0.0	0.0	6.7	1.3
4	3.1	3.6	0.0	2.7
5	84.4	17.8	0.0	42.7
7	9.4	7.1	6.7	8.0
8	3.1	42.9	0.0	17.3
9	0.0	28.6	86.7	28.0



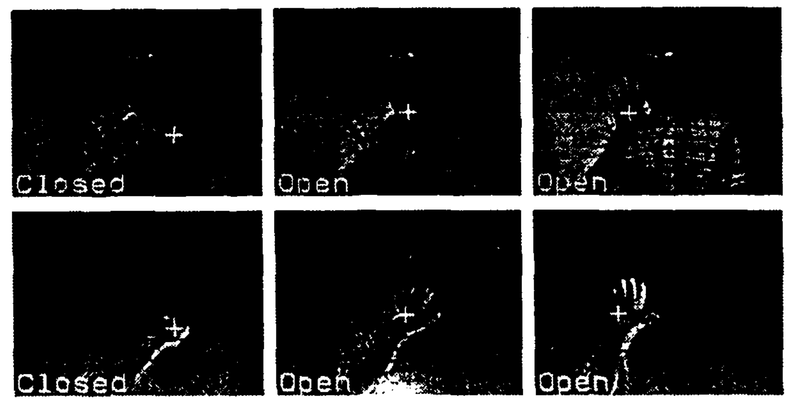


Hand Tracking

Problem: Tracking a hand through a sequence of video frames.

Two sources of input:

1. Intensity images
2. Difference images



Network Architecture

Two experts, each 3 layer convolutional networks:

1. Receives only intensity image as input.
2. Receives only difference image as input.

Gating network:

- 3 layer network, convolutional hidden, non-convolutiona output
- low resolution versions of intensity and difference images as input
- simple global features can effectively weight two experts

Extensions

1. Hierarchical Decomposition – Trees of Experts (Jordan & Jacobs 92)

$$P(d|x) = \sum_i g_i \sum_j g_{j|i} P(d|x, E_{ij})$$

– EM based IWRLS algorithm for linear experts

2. Support regions for multiple hypotheses
– one set of experts
– multiple gating networks
– multiple weighted averages of expert outputs

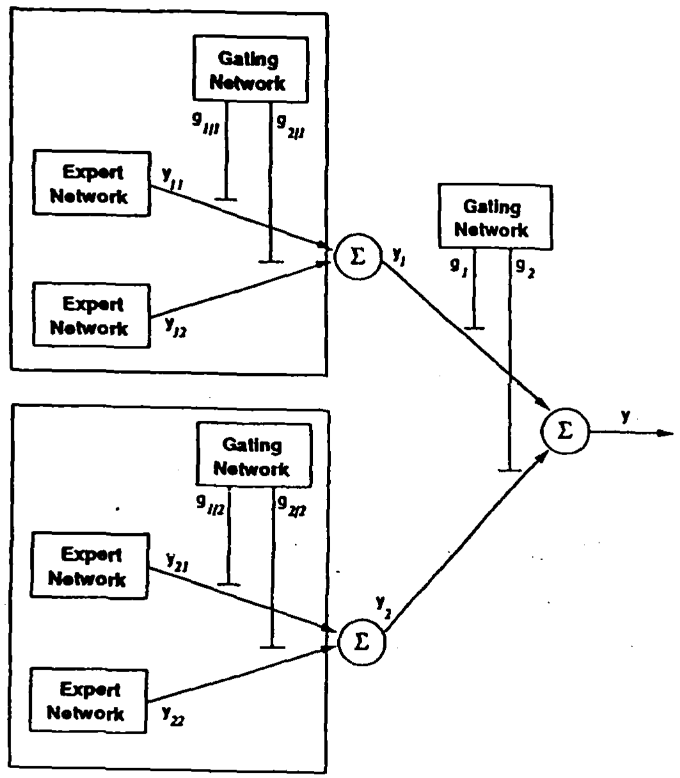
$$L = \prod_h P(d_h|x)$$
$$P(d_h|x) = \sum_i g_{hi} P(d_h|x, E_i)$$

3. Gating based on state trajectories
– HMM like extension to M.E.

$$L(y_1 \dots y_t | x_1 \dots x_t) =$$

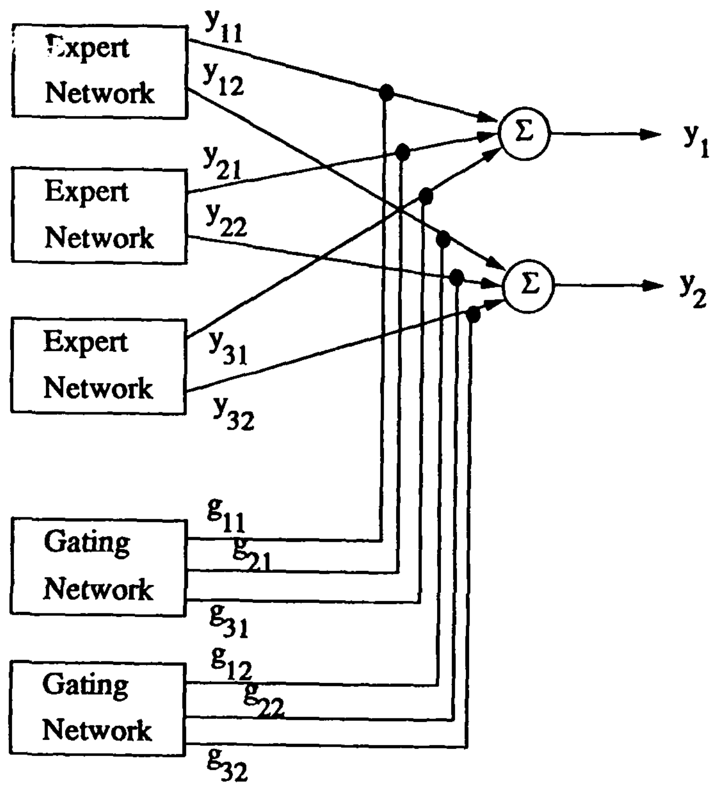
$$\prod_t \sum_k P(y_t | s_t = k, x_t) P(s_t = k | y_1 \dots y_{t-1}, x_1 \dots x_t)$$

Hierarchy of Experts



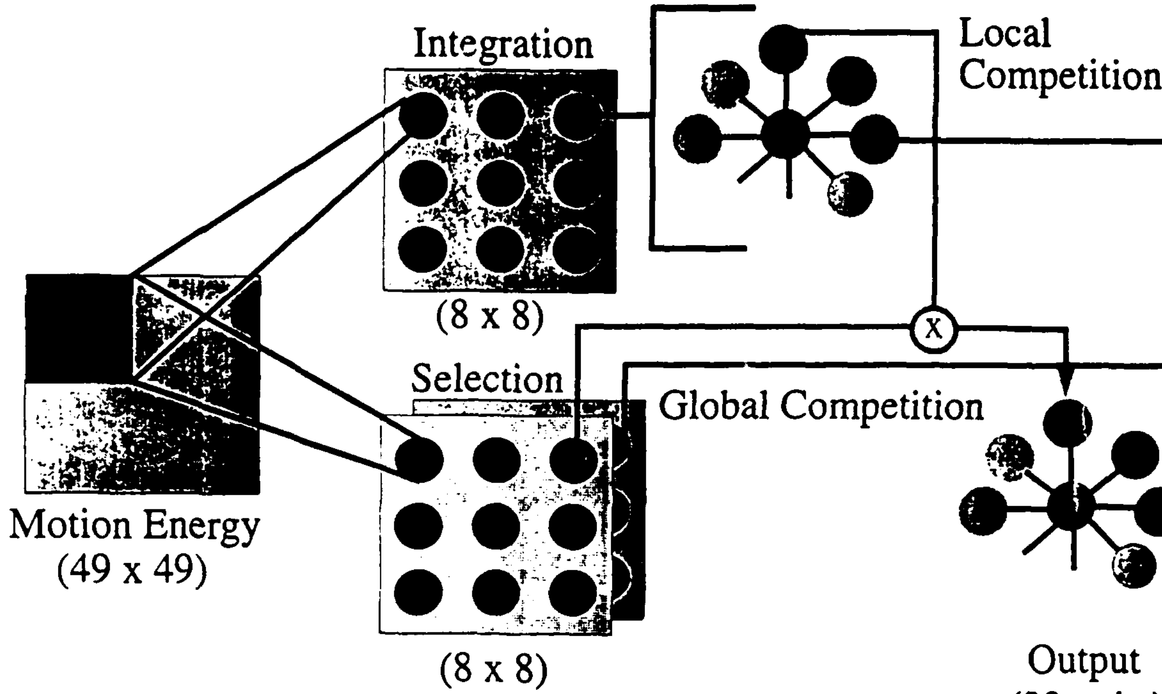
1. Error for expert ij: $E_i = p(b_i|\vec{d}_c)p(b_{ij}|\vec{d}_c)(\vec{d}_c - \vec{o}_i)$
2. Errors for gating net ij: $E_g(x_{ij}) = p(b_{ij}|\vec{d}_c) - g_{ij}$
3. Errors for gating net i: $E_g(x_i) = p(b_i|\vec{d}_c) - g$

Multiple Support Sets



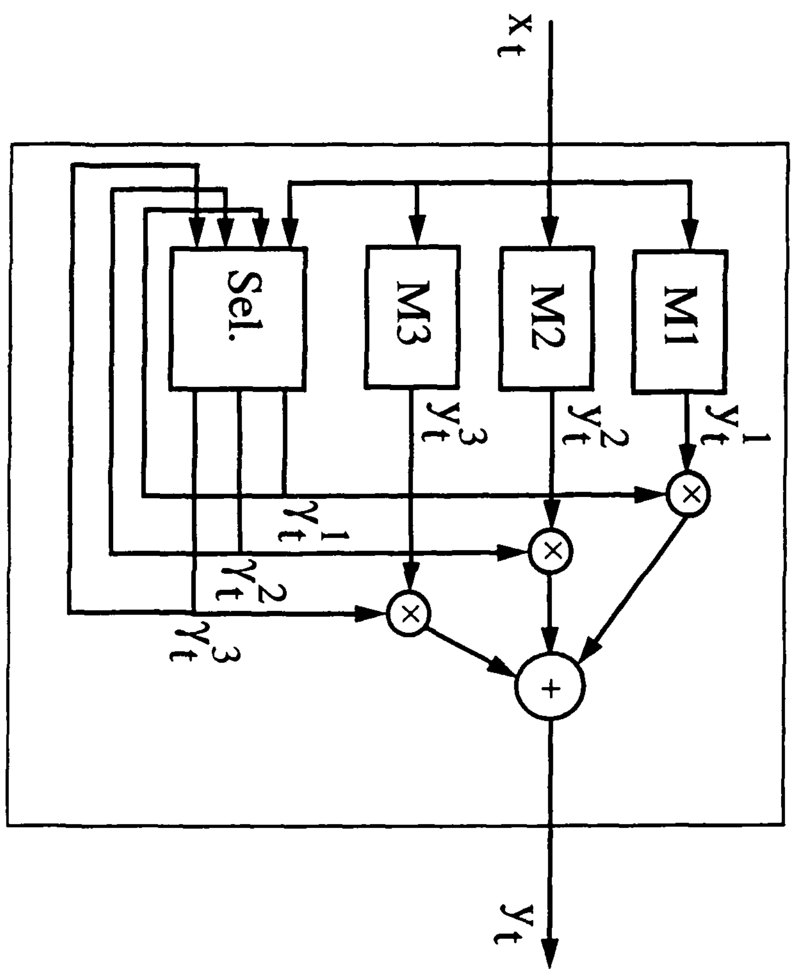
1. Error for expert y_{ij}: $E_{ij} = p(e_i|d_j)(d_j - y_{ij})$
2. Errors for gating net g_{ij}: $E_g(x_{ij}) = p(e_i|d_j) - g_{ij}$

INTEGRATION AND SELECTION



Trained -- Dual to mixture of experts:

$$\log L = \sum_t \sum_k \log \left(\sum_{x,y} S_k(x,y) \cdot exp(-(v_k - I_k(x,y))^2) \right)$$



$$L(y_1,\ldots,y_t|x_1,\ldots,x_t) = \prod_t \sum_k P(y_t|s_t=k,x_t)P(s_t=k|y_1,\ldots,y_{t-1},x_1,\ldots,x_t) = \prod_t \sum_k b_t^k \gamma_t^k$$

$$P(y_t|y_1,\ldots,y_{t-1},x_1,\ldots,x_t) = \sum_k b_t^k \gamma_t^k$$

$$b_t^k = f_{w_k}(x_t) \qquad \gamma_t^k = f_{w_s}(x_t, \{\gamma_{t-1}^j\})$$

Update Equations

Models / Controllers

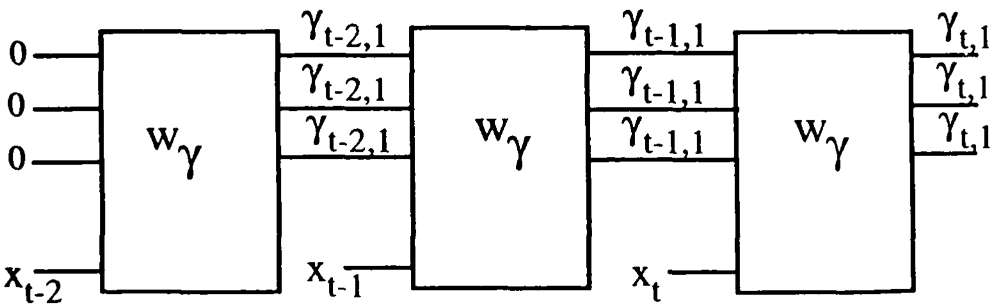
$$b_t^k = \frac{1}{\sqrt{2\pi}\sigma} e^{-(y_t - y_t^k)^2/2\sigma^2} \qquad L_t = \sum_k b_t^k \gamma_t^k$$

$$\frac{\partial \log L}{\partial w_k} = K \sum_t \underbrace{\frac{b_t^k \gamma_t^k}{L_t}}_{\beta_t^k} (y_t - y_t^k) \frac{\partial y_t^k}{\partial w_k}$$

Selector

$$\gamma_t^k = \frac{\exp g_t^k}{\sum_j \exp g_t^j}$$

$$\frac{\partial \log L}{\partial w_\gamma} = \sum_t (\beta_t^k - \gamma_t^k) \left(\sum_{\tau=0}^{T} \sum_j \frac{\partial g_t^k}{\partial g_{t-\tau}^j} \frac{\partial g_{t-\tau}^j}{\partial w_\gamma} \right)$$



Open Problems

1. Can theoretical results on averaging be extended to schemes like the M.E.?

2. Are there dynamic equivalents of stacking and boosting?

3. Hybrid architectures
   i.e. Basis Fn gating + MLP experts

BOOSTING

AND OTHER ENSEMBLE METHODS

HARRIS DRUCKER
CORINNA CORTES
L.D. JACKEL
YANN LECUN
VLADIMIR VAPNIK

AT&T BELL LABS

QUESTIONS ASKED:

FOR A GIVEN TRAINING SET SIZE:
WHICH ALGORITHM IS BEST ?

FOR A GIVEN COMPUTATIONAL COST:
WHICH ALGORITHM IS BEST?

WE WILL SHOW THAT

BOOSTING IS THE BEST ALGORITHM

GENERALIZATION IS BETTER IF WE TRAIN ON A SUBSET OF THE TRAINING POOL AND DISCARD THE REST

ADDITION IS BETTER THAN VOTING

figure-2.1a.talk

TRAINING SET

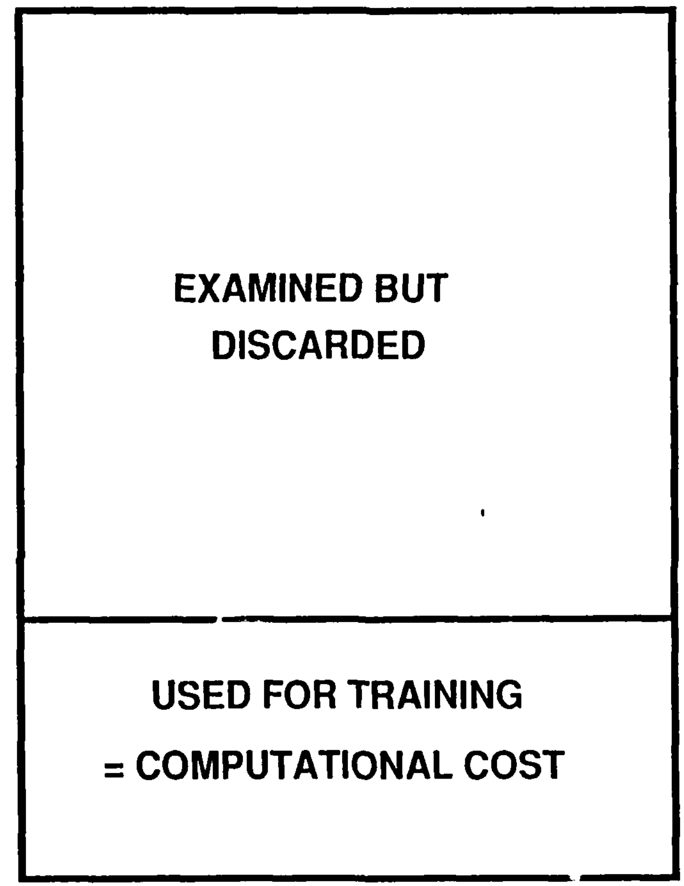


figure-2.3a

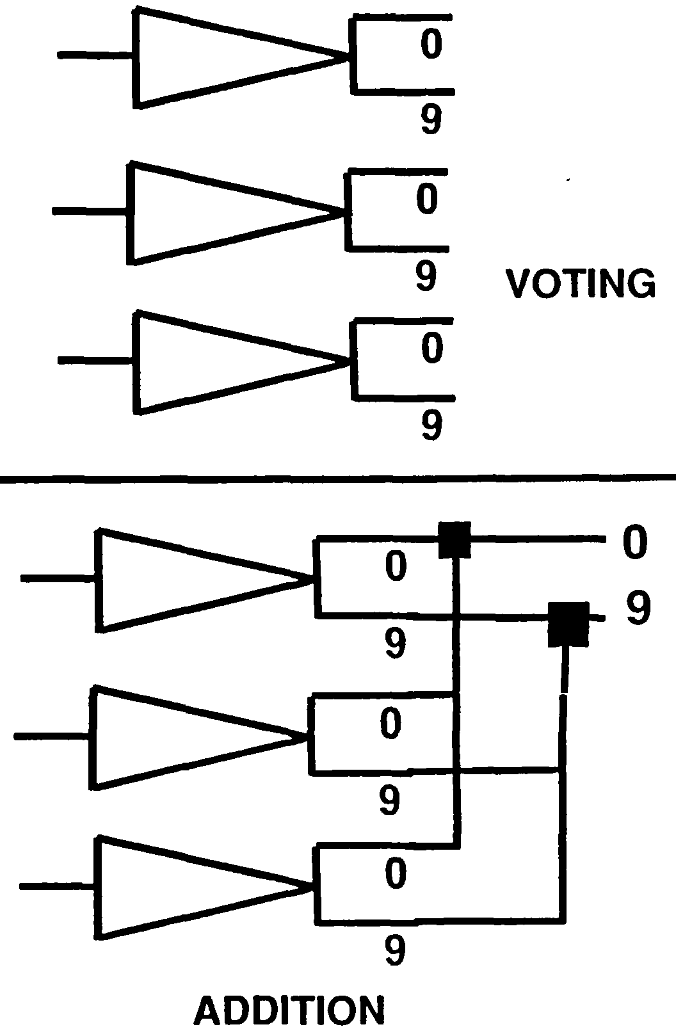


figure-2.4a.talk

DATABASE:

120,000 DIGITS FROM NIST
2000 FOR TESTING
10X10 INPUT SPACE

```
FOR  each algorithm (4 of them)
  FOR each architecture (3 of them)
    FOR each training set size
      REPEAT 10 times
        pick training set
        randomize weights
        train to mse minimum
      END_repeat
      get average test and train error
    END_FOR
  END_FOR
END FOR
```

figure-3a.talk

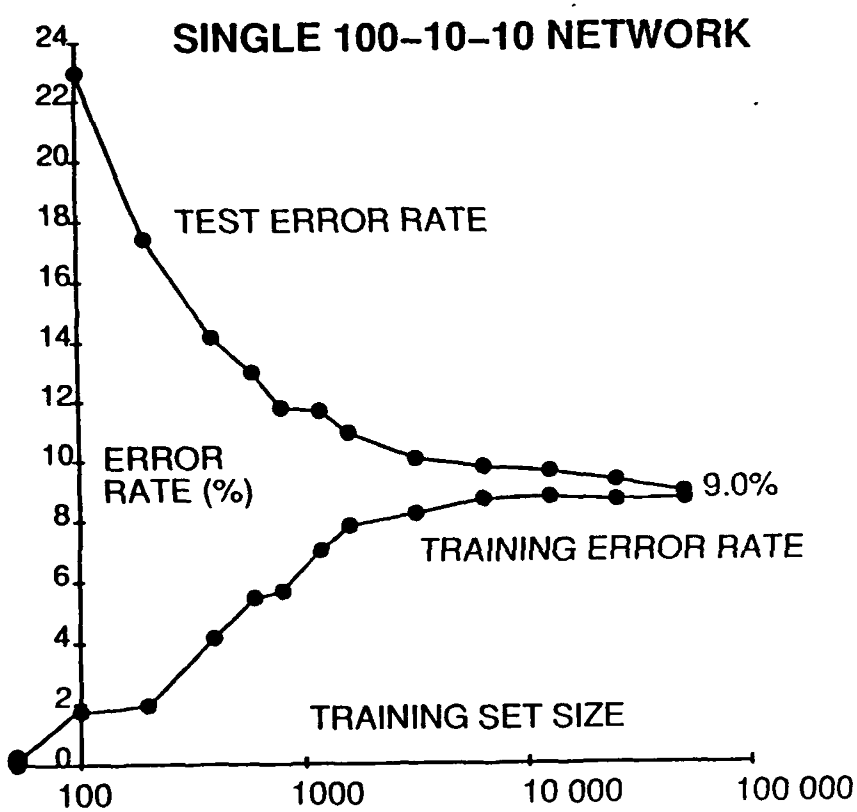


figure-4a.talk

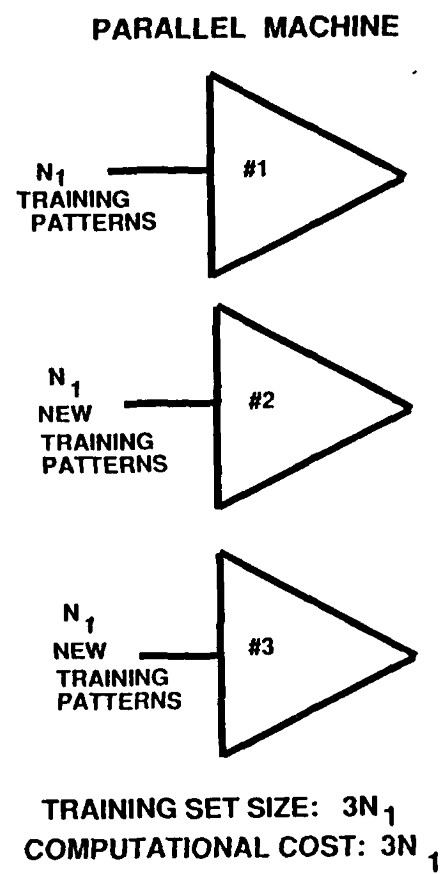


fig-5.talk

ORIGINAL BOOSTING

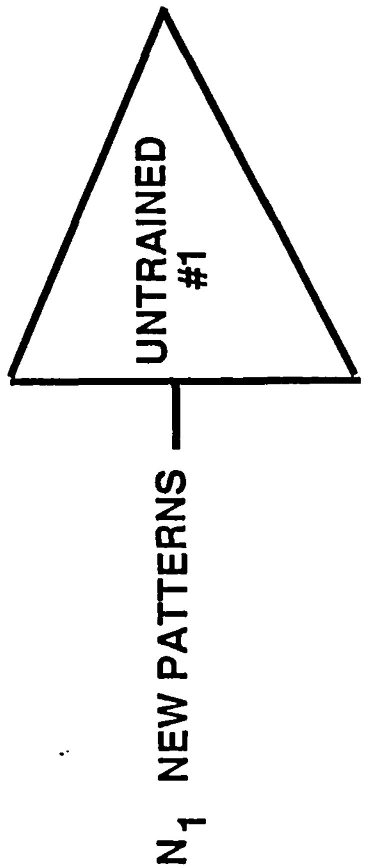


fig-6.talk

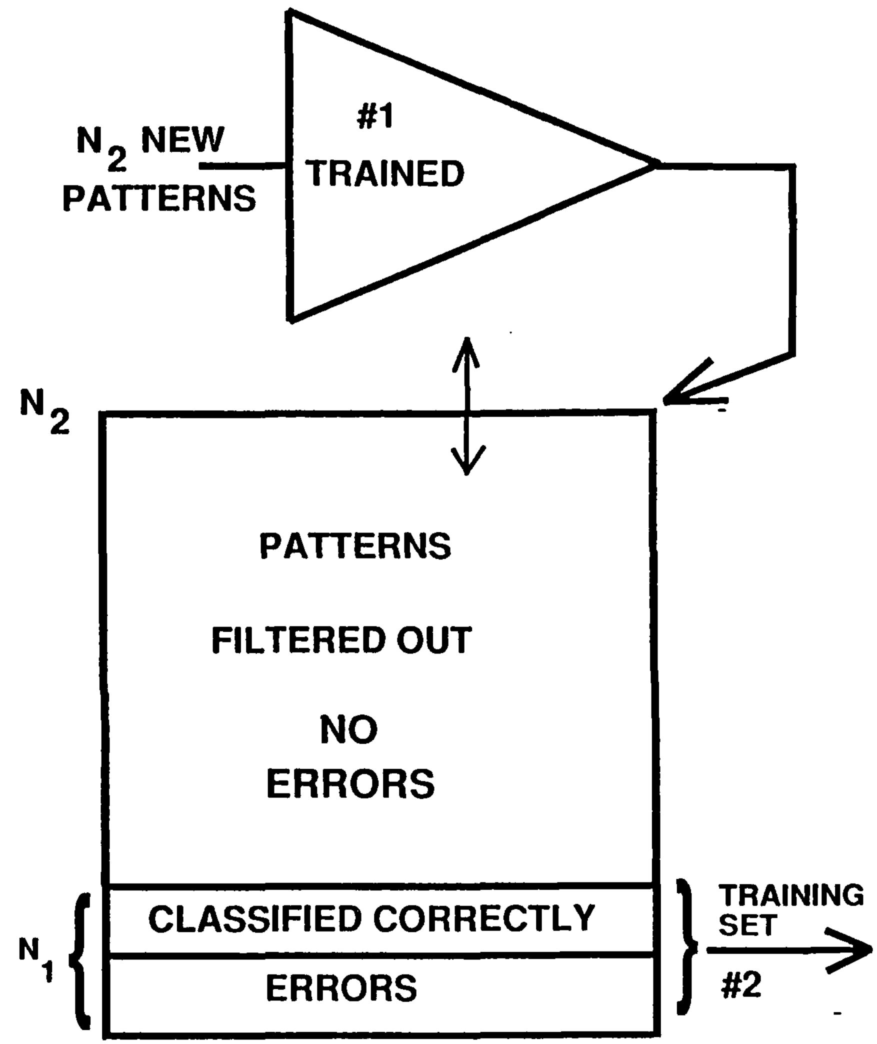


figure-7a.talk

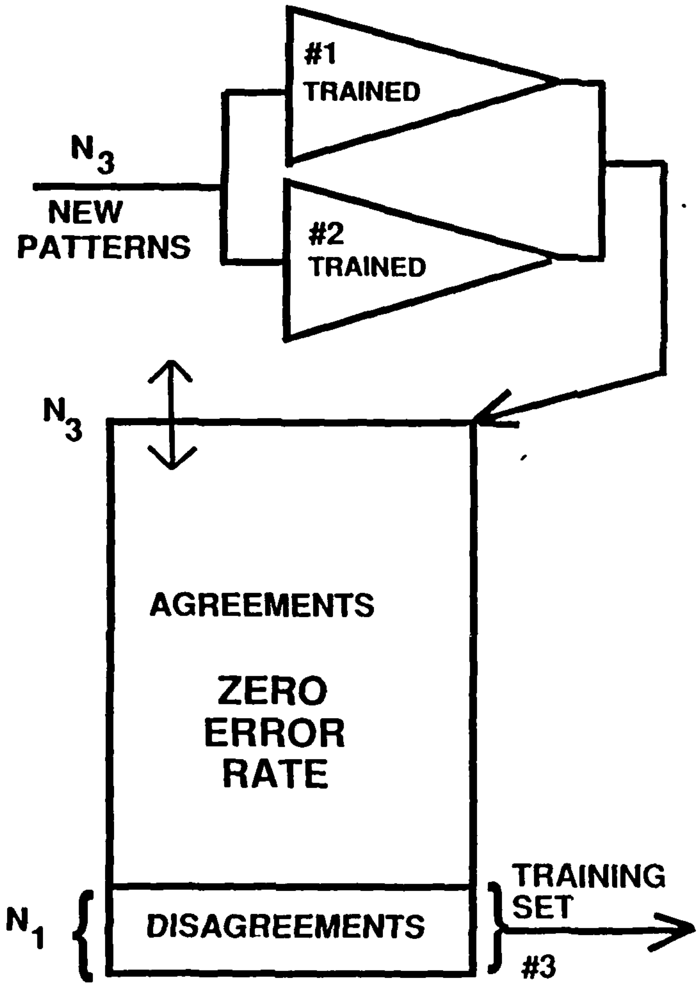
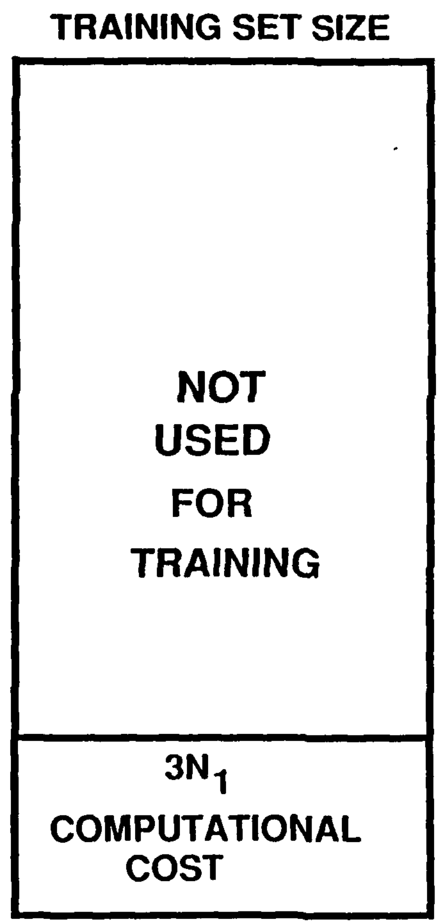


figure-8a talk



ORIGINAL BOOSTING

figure-9a.talk

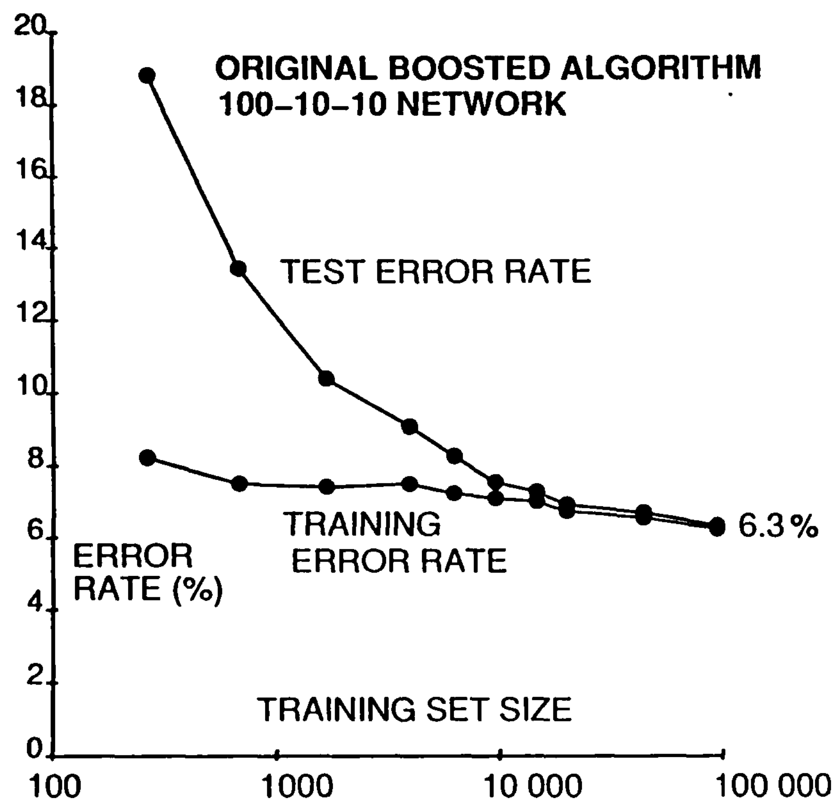


figure-10.talk

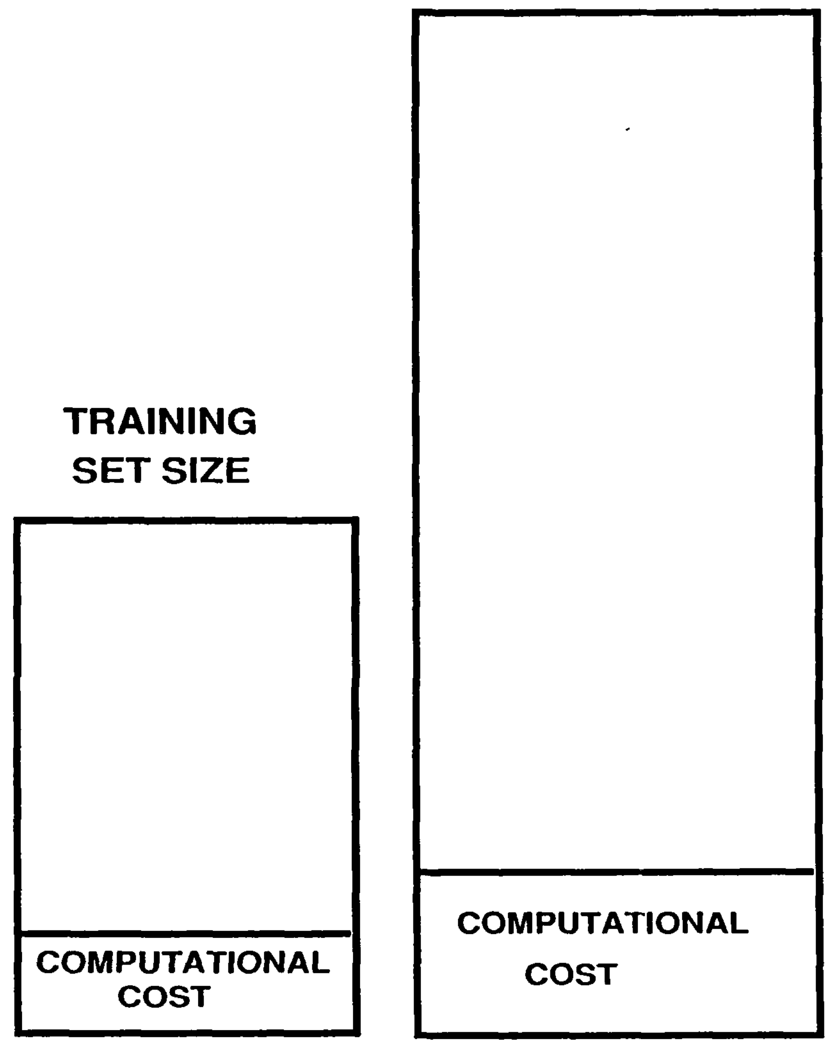


figure-11.talk

MODIFIED BOOSTING

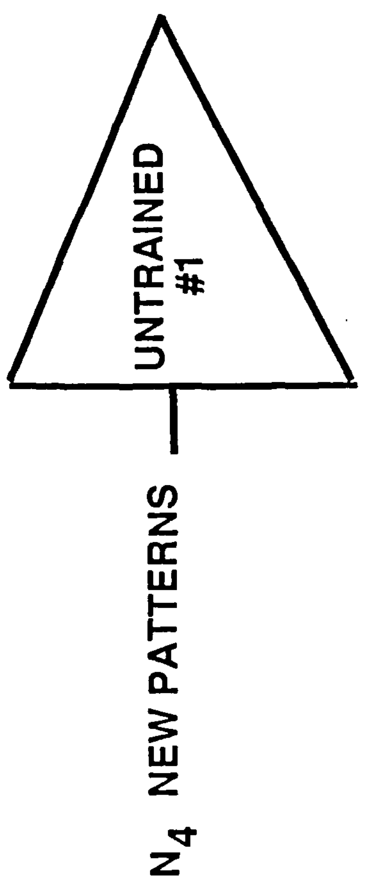


figure-12.talk

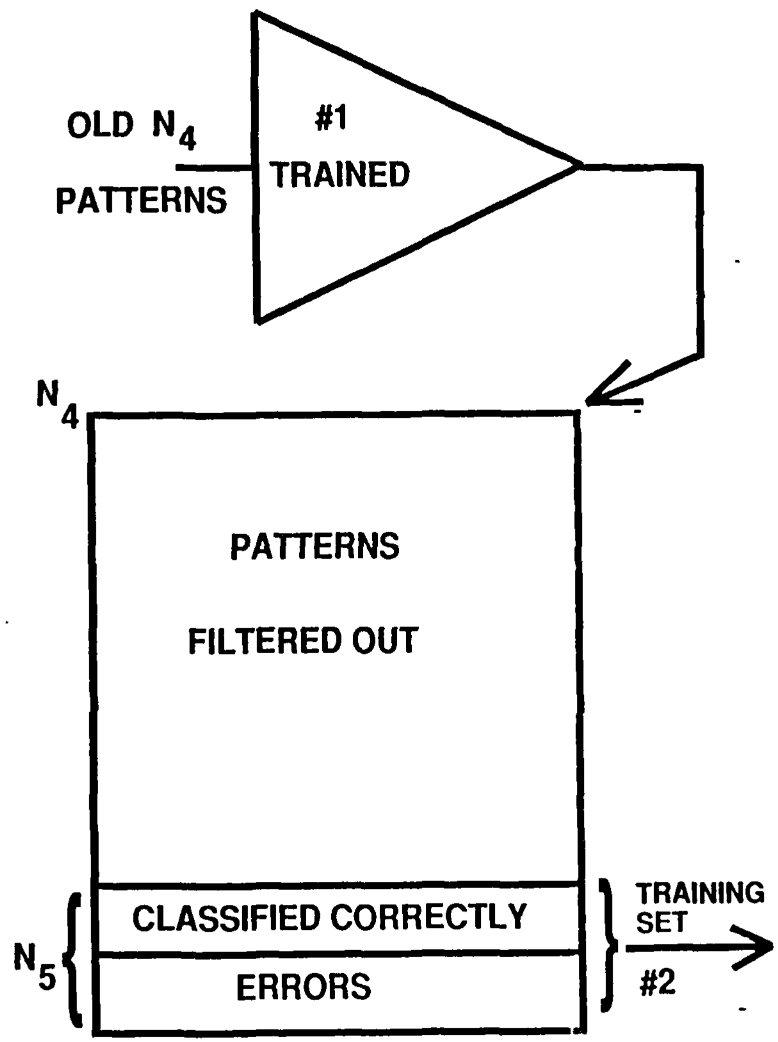


figure-13.talk

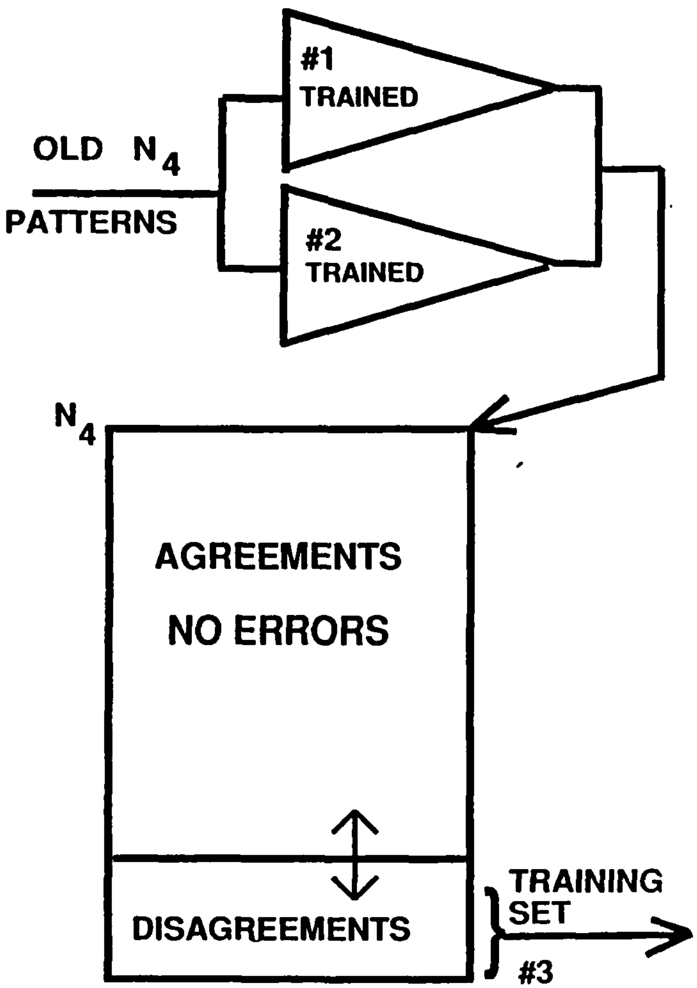
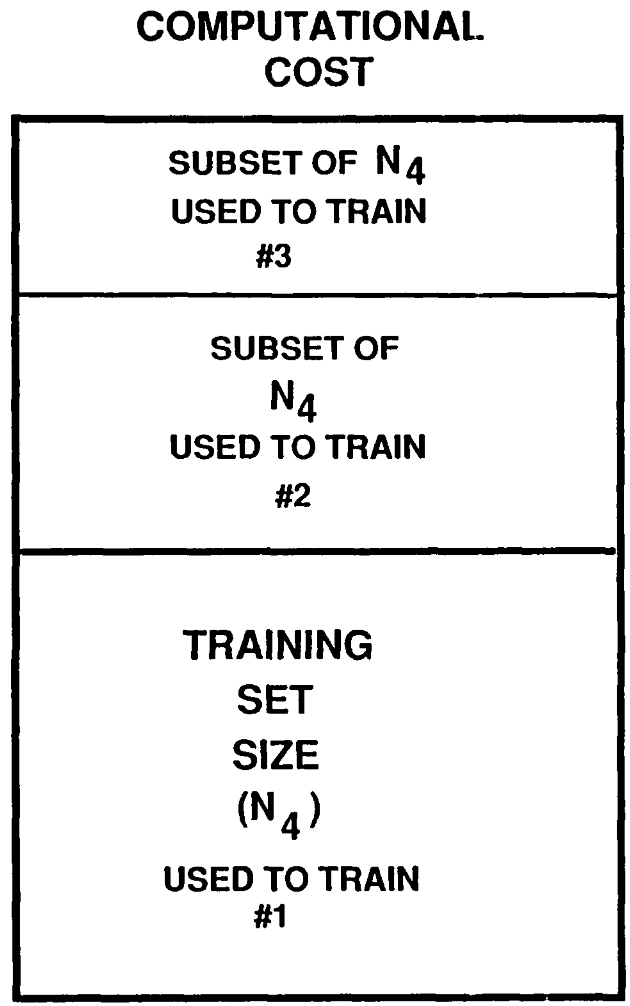


figure-14a.talk



MODIFIED BOOSTING

figure-15a.talk

100–10–10 NETWORK

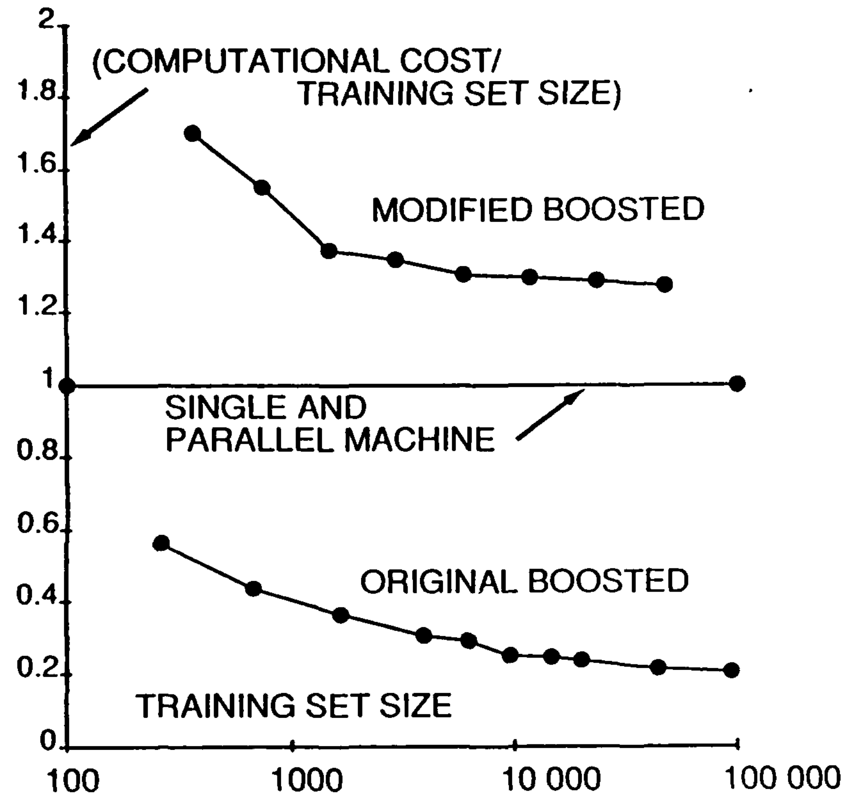


figure-16.talk

100–10–10 NETWORK

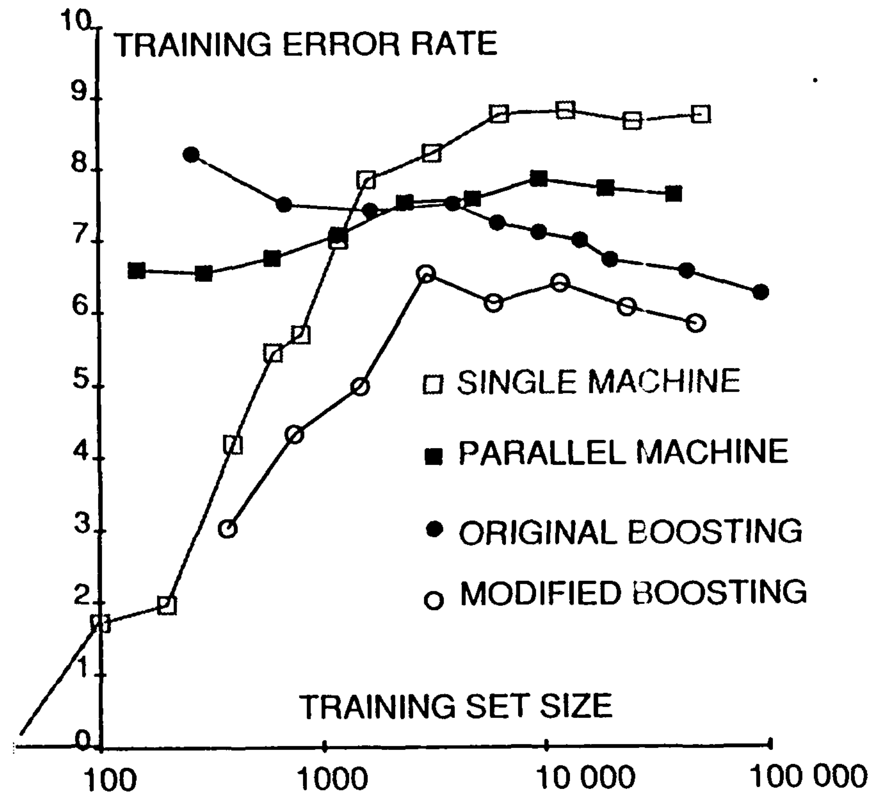


figure-17.talk
figure-17a.talk

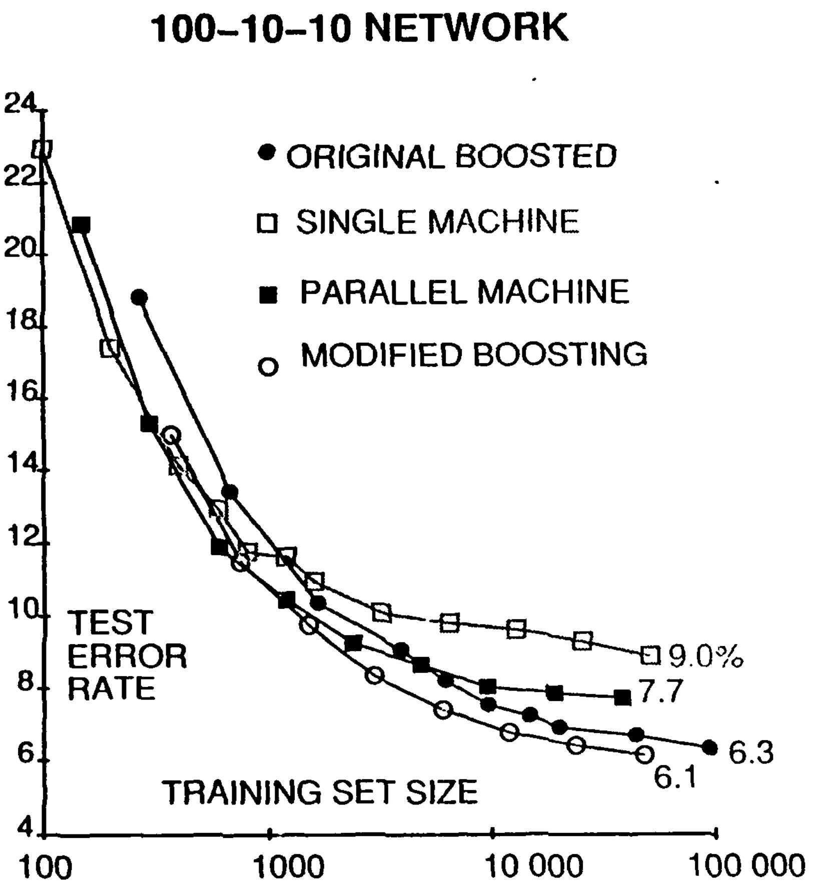


figure-18a.talk

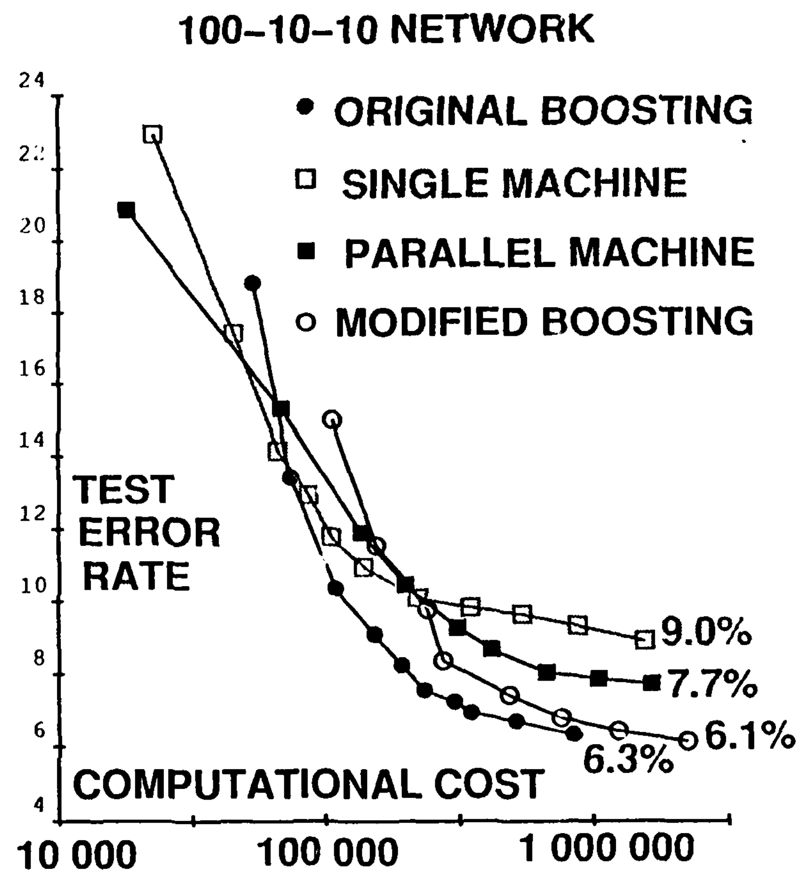


figure-20a.talk

100–10–10 NETWORK

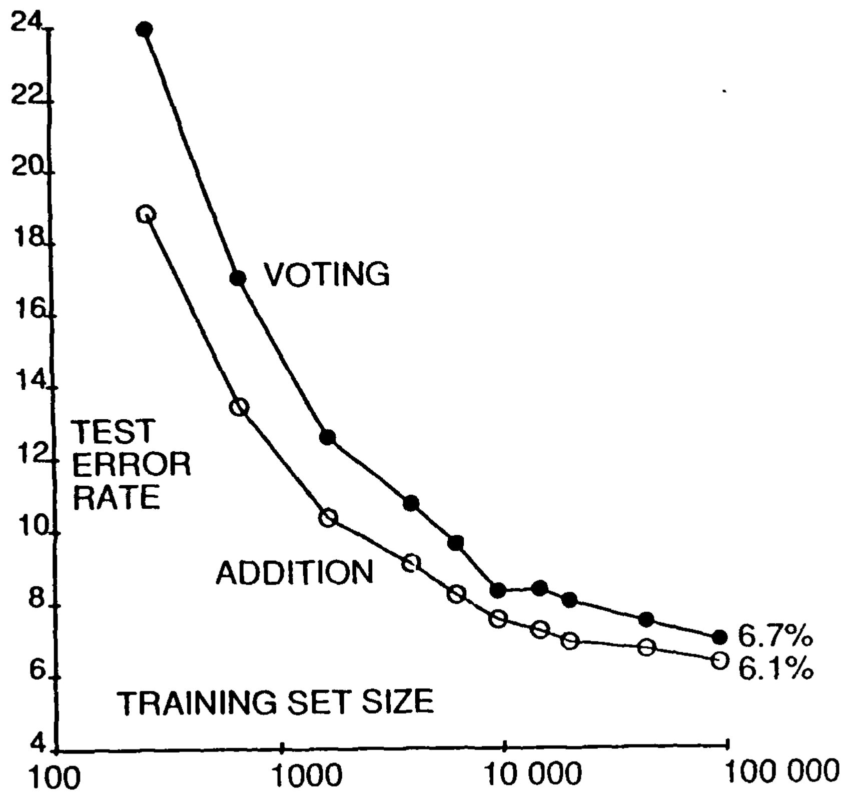


figure-21.talk
figure-21a.talk

A LARGER NETWORK
13,561 NEURONS
17,132 WEIGHTS
260,180 CONNECTIONS

TRAINING SET SIZE: 2.5 MILLION

COMPUTATIONAL COST: 180,000

ONE NETWORK: 113 ERRORS/10,000

BOOSTED ENSEMBLE: 70 ERRORS/10,000

figure-27a.talk

WE SHOWED THAT

BOOSTING IS THE BEST ALGORITHM

GENERALIZATION IS BETTER IF PART OF THE TRAINING POOL IS DISCARDED

ADDITION IS BETTER THAN VOTING

FOR THE ORGINAL BOOSTED ENSEMBLE THE TRAINING ERROR RATE DECREASES WITH INCREASING TRAINING SET SIZE

REFERENCES:

SCHAPIRE: THE STRENGTH OF WEAK LEARNABILITY, *MACHINE LEARNING,*

VOL 2, NO. 2, 197–227

DRUCKER, SCHAPIRE, AND SIMARD:
IMPROVING PERFORMANCE IN NEURAL NETWORKS USING A BOOSTING ALGORITHM,

NIPS 5, 42–49

figure-28a talk

A Commercial Application: Extracting Document Content from Images

Christopher L. Scofield
Harry Chang
Ed Collins

Nestor, Inc.

Structure of the Problem

- OCR is not really a problem of character recognition. It is really language processing from images:

Character context drives segmentation:

Lexical context drives character interpretation:

Nestor, Inc.

Structure of the Problem

Lexical context drives character interpretation:

clone

clone?
done?

Application specific rules drive interpretation:

54Y293

54Y293?
54Y2A3?
J4Y2A3?

Nestor, Inc.

Structure of the Problem

Document structure drives syntactic and lexical possibilities:

VEGA RESOURCES
2931 LOVELANE
TINSEL WA 02931

Company Name

Street number Street name

City State Zip

Nestor, Inc.

What are the possible approaches?

- Single Network Architecture
  - » [Keeler90]: Combined segmentation and recognition;
  - » [Fontaine92]: RNN trained on pixel-columns
- Pluses:
  - » Makes no assumption about structure of problem
  - » Automatically trains each part of problem
- Minuses:
  - » Scaling problem
  - » Lack of modularity: application dependent

Nestor, Inc.

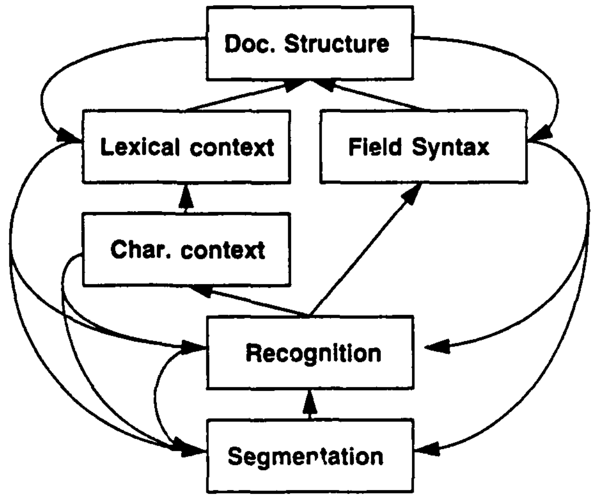
Possible methods

- Multiple Network Architecture
  - » [Gouin92]: Neural network segmentation for map processing;
  - » [Scofield92]: Context-driven segmentation, recognition
- Pluses:
  - » Each module can be built in a minimal fashion
  - » Only some parts need to be changed for new applications
- Minuses:
  - » Assumes prior knowledge
  - » Must be assembled in a piecewise fashion
  - » Credit assignment problem

Nestor, Inc.

A Multiple Network Approach to OCR for Handwriting

- Task decomposition:



Nestor, Inc.

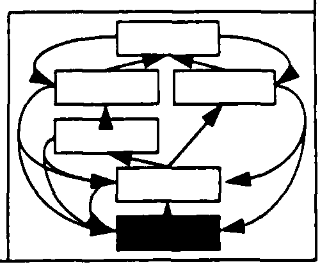
Neural Network Segmentation

- Neural network is used to assemble a tree representation of the image:

(1) Classify all blobs into "Character", "Noise", and "Mixed"

(2) Recursively segment "Mixed" until decomposed into only terminal nodes "Character" and "Noise"

(3) Compose a list of possible alternative segmentations
Fragmented characters
Optimal window adjustment

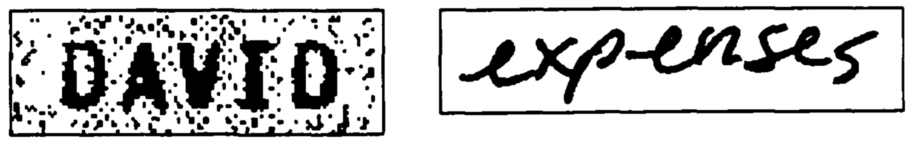


Nestor, Inc.

Segmentation: Step 1

- 3-layer BPN trained to classify blobs into:

"Character" "Noise" "Mixed"



- Use connectivity analysis features [Hu62] including:

area, perimeter, number of holes,
area of holes, principle moments,
aspect ratio, etc.

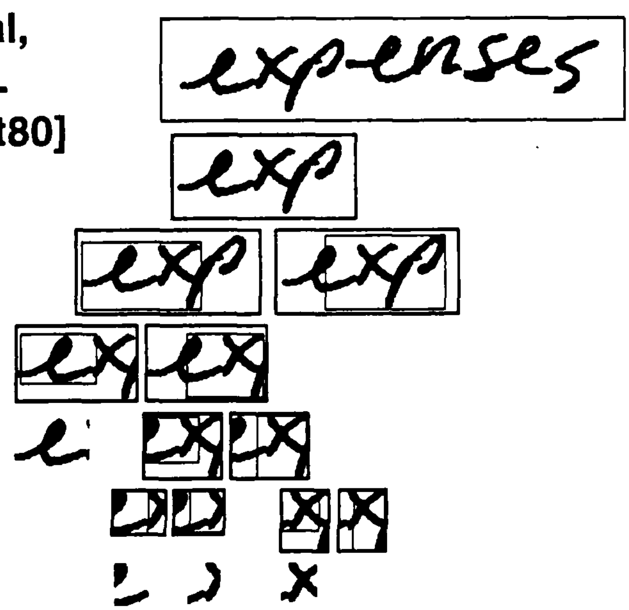
Nestor, Inc.

Segmentation: Step 2

- "C", "N" are terminal,

"M" parsed with quad-tree analysis [Samet80]

Re-classified at each step:



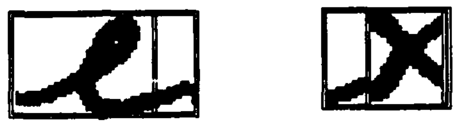
Nestor, Inc.

Segmentation: Step 2 (contd):

- Hierarchical Agglomerative Clustering [Duda72] groups terminal nodes; re-classified to ensure still terminal:

Step 3:

- List of segmentation alternatives compiled for classification into characters
  - Multiple "cuts" of characters provided for later analysis:



Nestor, Inc.

Segmentation Accuracy

- Test set consists of 5,654 HP/MP characters in 1,236 words (46% HP) selected from 53 real-world documents
- HP data consists of live forms with constrained HP, unconstrained HP, run-on HP and some cursive
- Character segmentation accuracy:

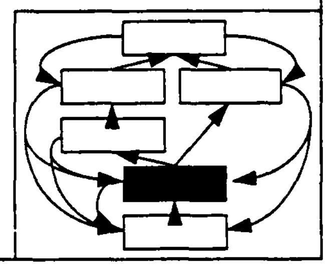
(First choice correctly segmented)

Segmentation Network	Correct	Incorrect
"Blob" features (7-10-3)	92.7	7.3

Nestor, Inc.

Character Recognition: Overview

- Segmentation alternatives are processed for character class
- Use two static feature sets: (cf - [LeCun90])
- Three-layer, feedforward BPNs are used as estimators of *a posteriori* probabilities
- We have employed three types of hybrid networks:
  - Glue networks [Waibel88]
  - Parallel Experts [Reilly87]
  - Hierarchical Filters [Reilly87]



Nestor, Inc.

Character Recognition: Feature Extraction

- Segmentation alternatives are converted to grey-level: gaussian kernel estimated from line widths
- Pixel Feature Set:
  - Pixel map is sub-sampled with grid producing coarse map (100-element grid)
- Edge Feature Set:
  - Edge-map produced from grey-level gradient estimation [Roberts65] (4 edge directions)
  - Edge map is sub-sampled with grid producing coarse edge map (30-element grid)

Nestor, Inc.

Character Recognition Classifiers

- **Features used to train 3-layer, feedforward BPNs**

Data Set	# Authors	Digits	Alpha U/LC
Train: NIST 1,3; Propr.	2600	265,000	120,000
Test: Propr.		4,767	12,932

- **In addition to using "Forced accuracy", can use a heuristic which models high cost of errors:**

  "Figure of Merit": FM = 100 - 10(%E) - %R

Numeric Network	FM	Correct	Inc.	Reject	Forced
Edge features (120-32-10)	95.15	97.04	0.21	2.75	99.01
Pixel features (100-45-10)	93.41	95.87	0.27	3.86	98.59

Nestor, Inc.

Classifier Analysis

- **Using "rule-of-thumb" e = W/T [Baum89]:**

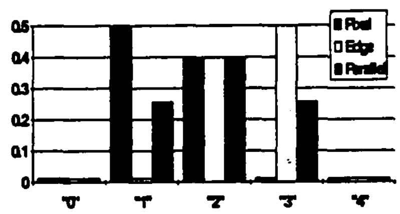
Training set:	T = 265,000
Edge Net:	W = 120*32+32*10 = 4,160
Expected test error:	e = 1.7%
Pixel Net:	W = 100*45+45*10 = 4,950
Expected test error:	e = 1.9%

Nestor, Inc.

Character Recognition: Parallel Experts

- How to combine the results from two networks?
- Could vote if have many “experts’. If only two, then average activation (probability) vectors:

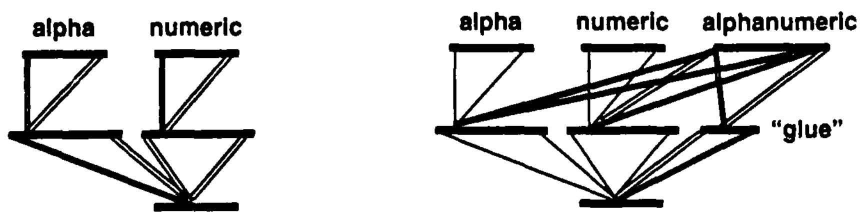
$$P_i = 1/2(P^e_i + P^p_i)$$



Numeric Network	FM	Correct	Inc.	Reject	Forced
Edge features (120-32-10)	95.15	97.04	0.21	2.75	99.01
Pixel features (100-45-10)	93.41	95.87	0.27	3.86	98.59
Parallel Nets:	97.25	98.20	0.10	1.70	99.39

Alphanumeric Character Recognition

- Support full alphanumeric HP
- Natural decomposition into alpha and numeric subnets
- Use glue-net architecture [Waibel88]:
  - Trained 3-layer nets for alpha (u/l case) and numeric
  - Freeze middle-layer weights, route activations to output
  - Add-in new “glue” layer to resolve inter-class ambiguity
  - Train second layer of weights and all glue-cell weights

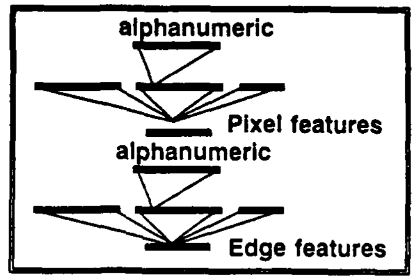


Nestor, Inc.

Alphanumeric Accuracy

- **Results:**

Network Architecture	FM	Forced Correct	Forced Incorrect
Numeric sub-net (120-32-10)	n/a	95.15	4.75
Alpha (U/L) sub-net (120-120-26)	n/a	91.41	8.59
Single Glue Net(1) (120-210-36)	51.24	89.24	10.76
Hierarchical (Super) Glue Net	54.04	90.80	9.20
Single Glue Net(2) (120-210-36)	54.46	89.98	10.02



Nestor, Inc.

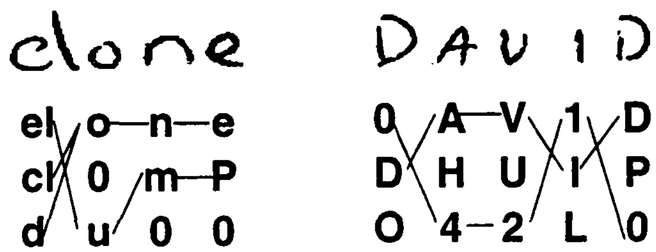
Glue Net Analysis

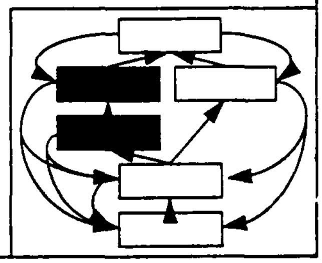
Digit set:	265,000
Digit Net:	W = 120*32+32*10 = 4,160
Expected test error:	e = 1.7%
Alpha set:	120,000
Alpha sub-net:	W = 120*120 + 120*26 = 18,720
Expected test error:	e = 15.6%
Full set:	385,000
Glue weights:	W = 120*58 + 210*36 = 14,520
Expected test error:	e = 3.8%

Nestor, Inc.

Character and Lexical Context

- Word recognition: determine the best string interpretation given all sources of knowledge:
  - segmentation alternatives
  - character recognition probabilities
  - character transition probabilities
  - lexical context





Nestor, Inc.

Character and Lexical Context

- Use the Viterbi algorithm [Viterbi67, Forney73] to select the character string with maximum *a posteriori* probability
- Let maximization of word probability drive proper segmentation [Bozinovic82]

- Problem: VA can produce lexically incorrect strings. Post-processing with a dictionary can produce word which is not MAP.
- Solution: Use lexical context to trim paths from VA search ensuring that the final string is both MAP and lexically correct [Srihari83].

Nestor, Inc.

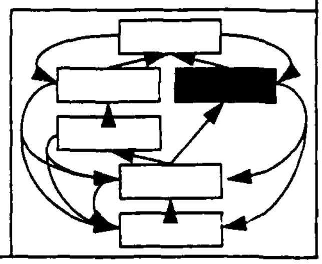
Application Context Processing

- Some applications are alphanumeric but not part of lexicons: **AIB00I38**
  - Inter-character statistics are specialized, hard to learn without large set
- User-definable syntax selects which subnet to use for each character position, trims segmentation alternatives to match syntax

22152

ZIP Code → 2 2 1 5 2

2 4 1 5 2



Nestor, Inc.

Status and the Future

- This architecture is the basis for the product NestorReader
- To be supported on Ni1000 neural net chip
- Extensible to character-based cursive recognition
  - Now developing much larger training and test sets for run -on HP and cursive
  - Developing stats. on character and segmentation accuracy due to character and lexical context

Nestor, Inc.

From talks presented in June 1993 at the International Forecasting Symposium, Pittsburgh and the Financial Economics Workshop, Santa Fe Inst.

Forecasting the U.S. Index of Industrial Production via Neural Networks

John Moody,
Uzi Levin, & Steve Rehfuss
Oregon Graduate Institute
moody@cse.ogi.edu

Outline

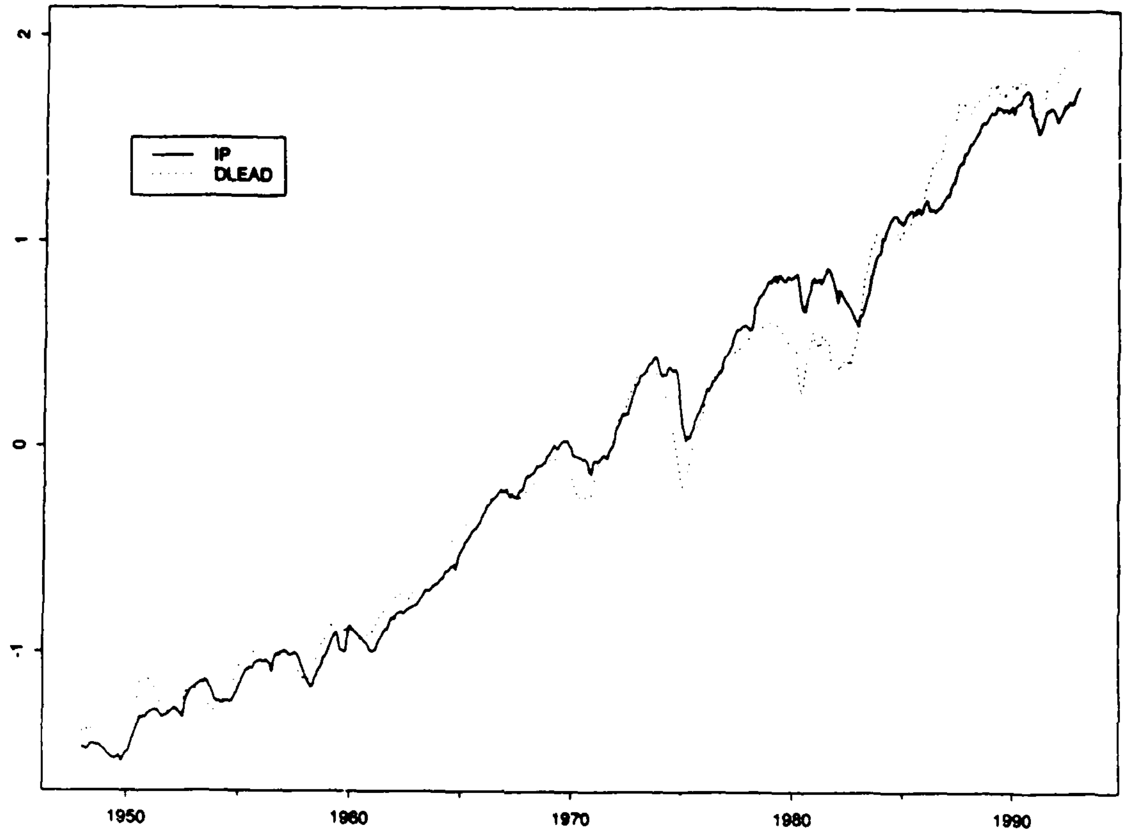
I. Intro & Statement of Problem
   - Performance of Professional Economists
   - Difficulties:
     - Non-Normality
     - Non-Linearity
     - Non-Stationarity

II. Performance of Linear Models
   - Univariate AR Models
   - Multivariate Regression & ARX Models

III. Results for Multivariate NN Models
   - Regression Nets
   - Classification Nets
   - The "Noise/Nonstationarity" Trade-Off
   - Committees

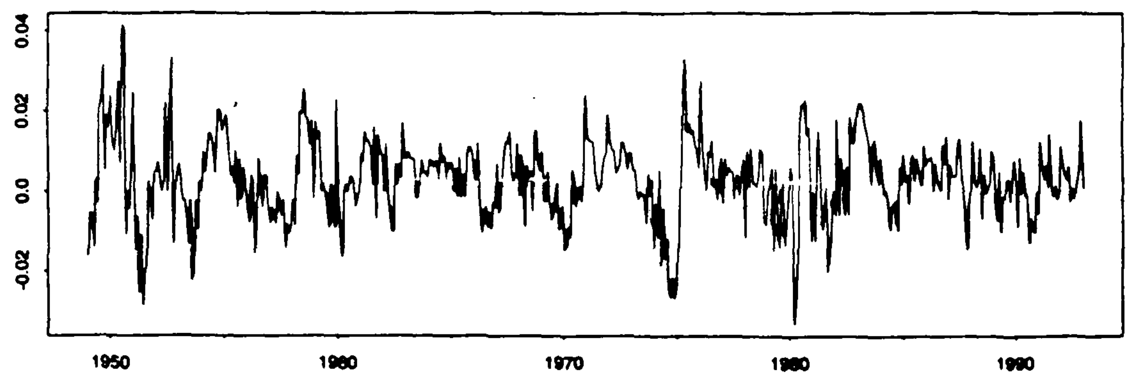
IV. Results for SPX

SCALED IP & DLEAD INDICES

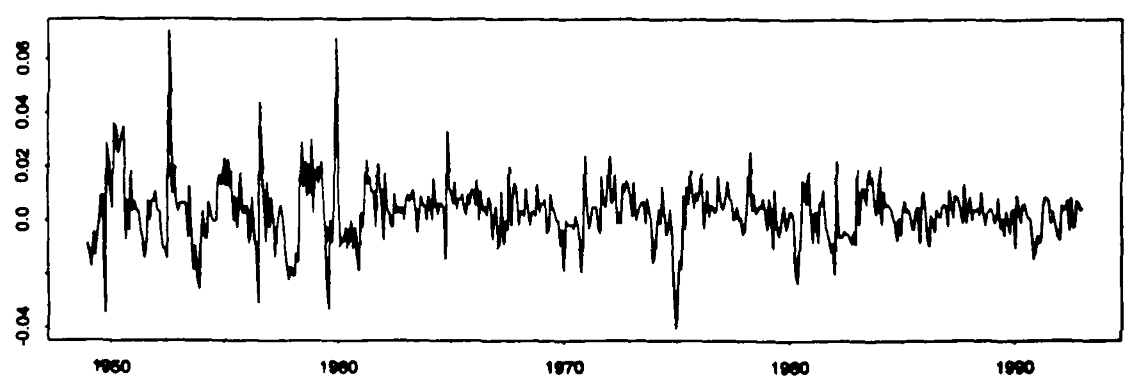


One month rates of returns

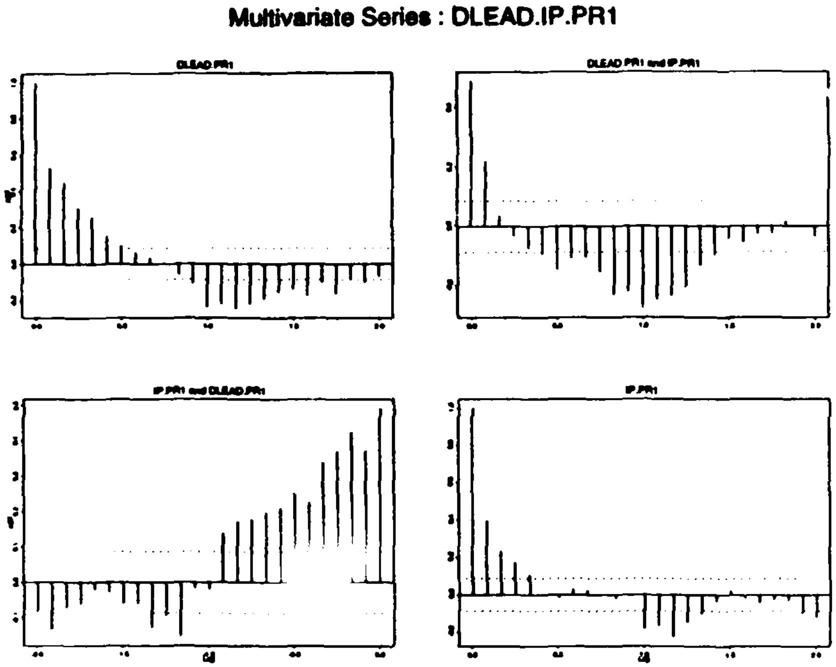
DLEAD.PR1



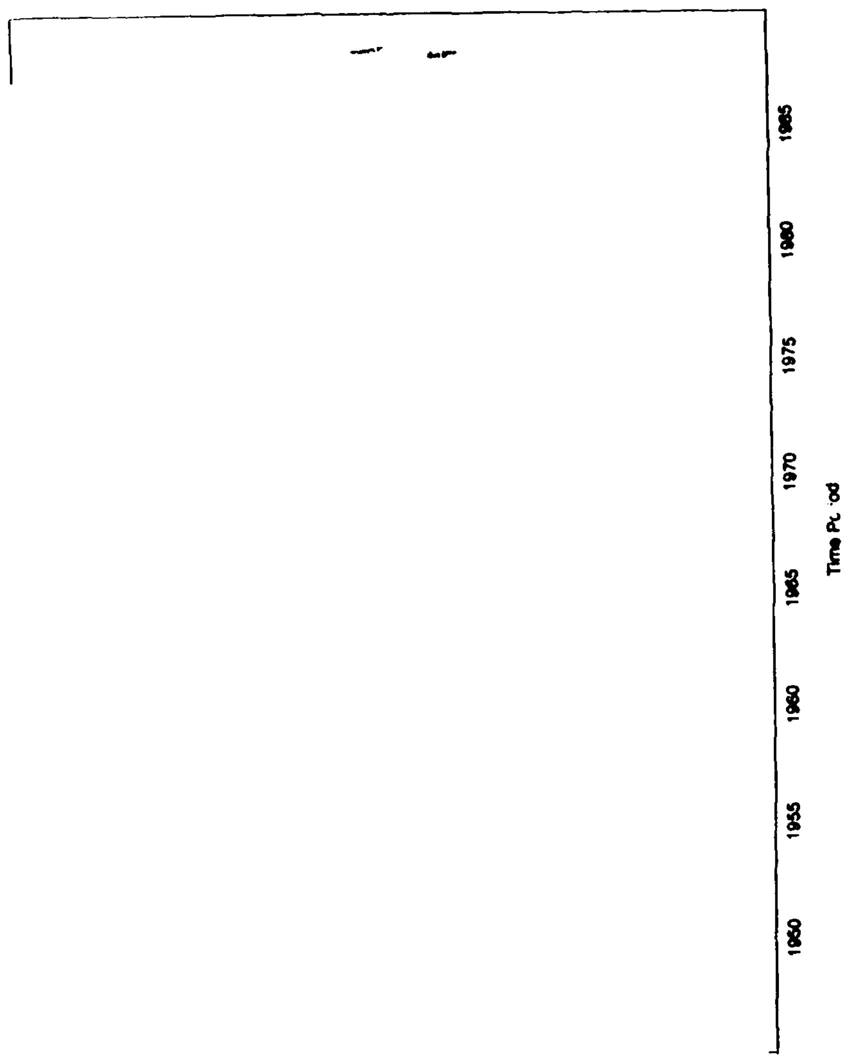
IP.PR1

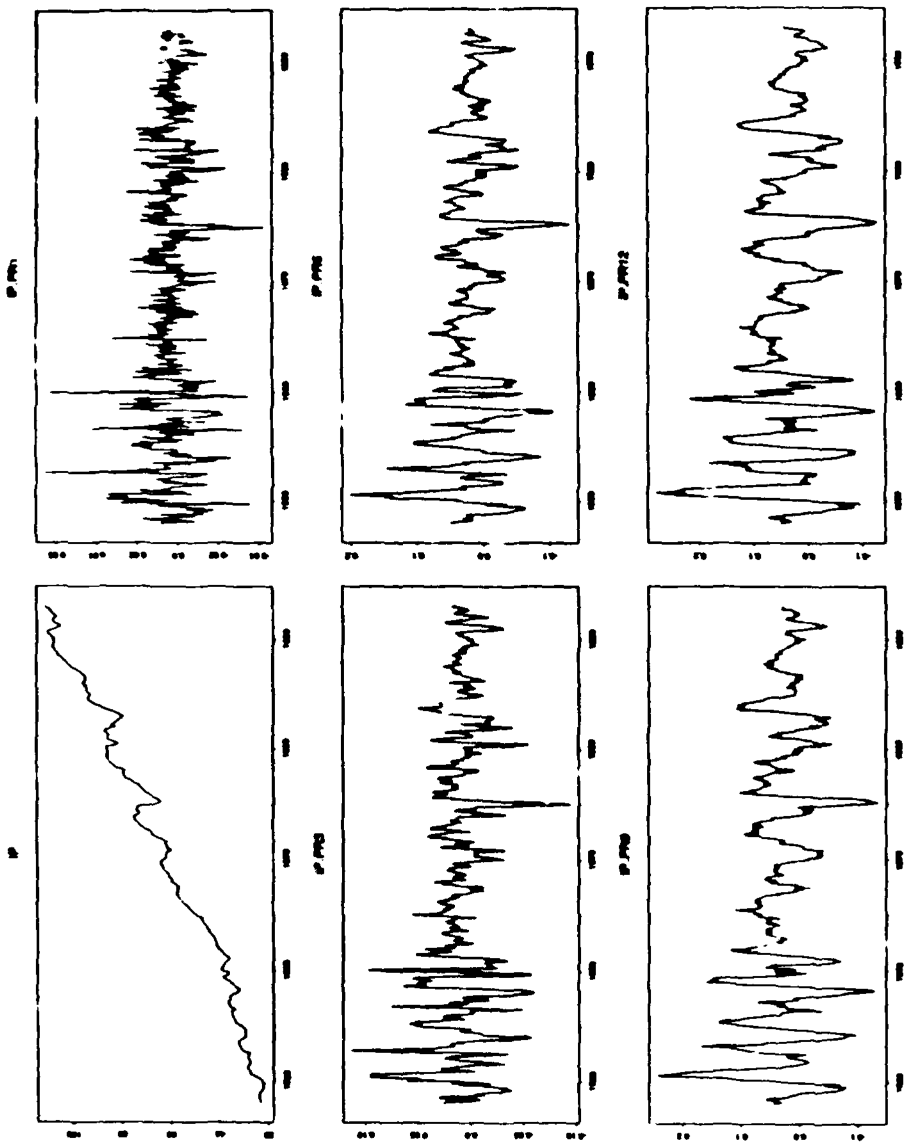


The leading index leads industrial production.



Note the nonstationarity of $\nabla \log(IP)$





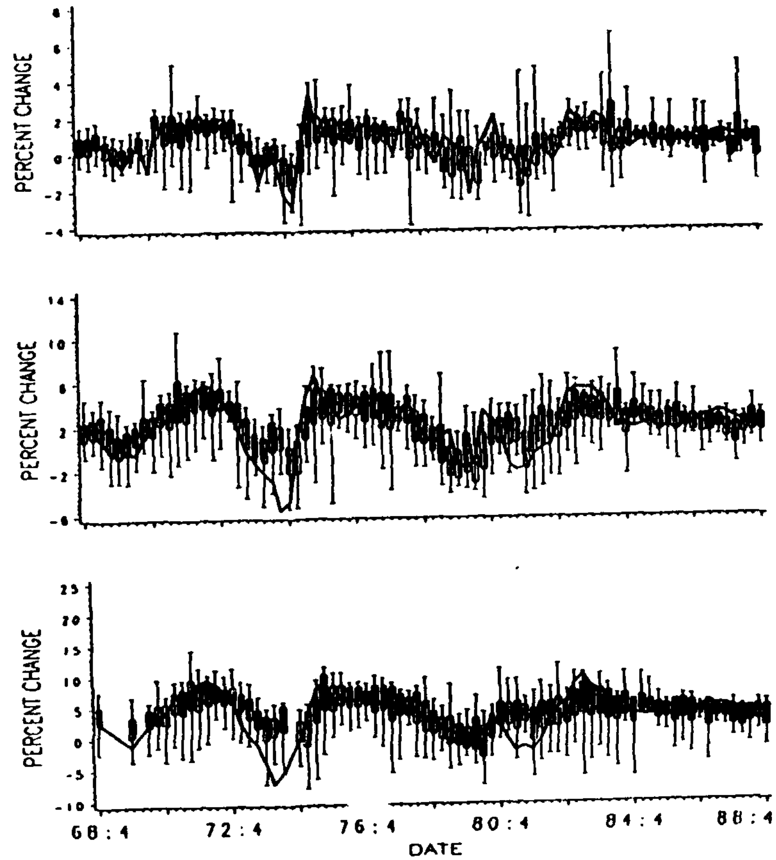
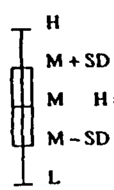


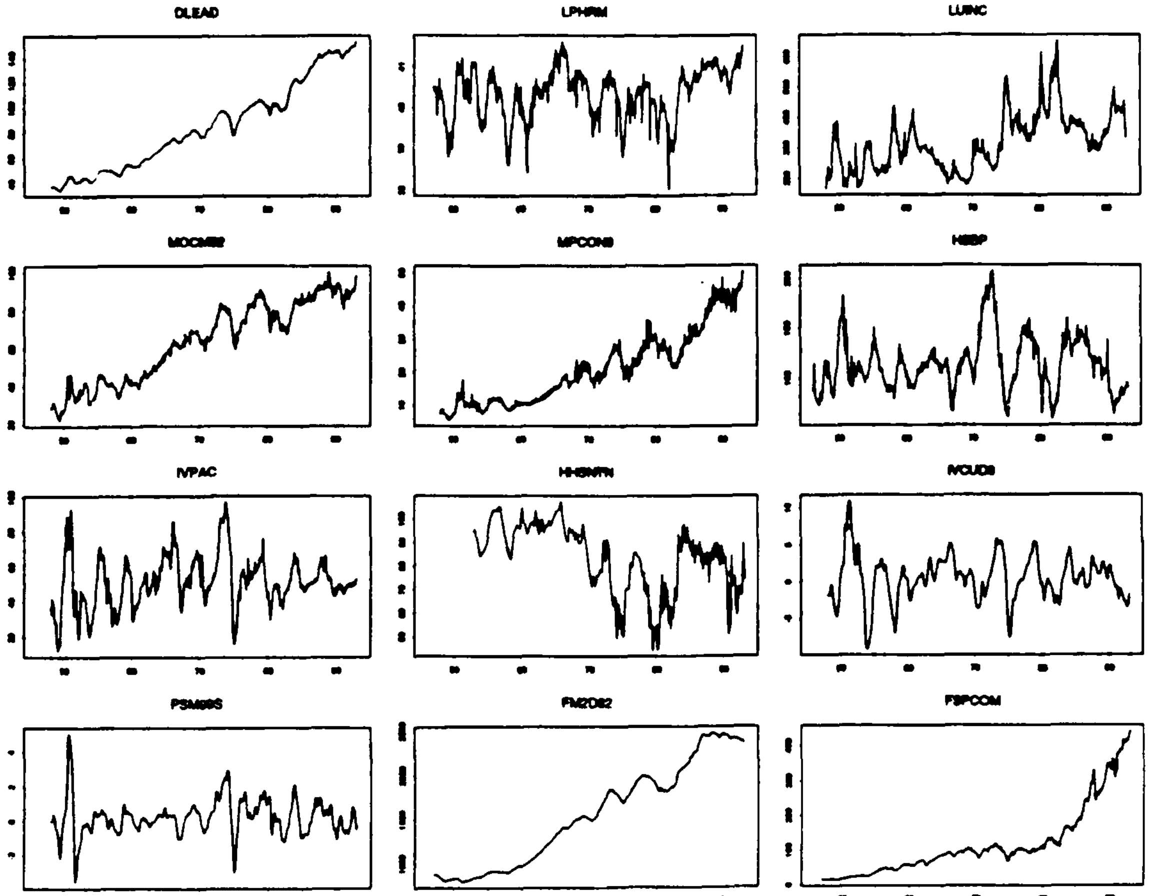
Fig. 1.1 Forecast distributions and actual values of percentage changes in real GNP, three horizons, 1968:IV–1990:I



H = high; M = mean; SD = standard deviation; L = low.

Source: Zarnowitz & Braun 1993

Note the poor quality of professional economists' forecasts.



The Index of Leading Indicators and its Components.

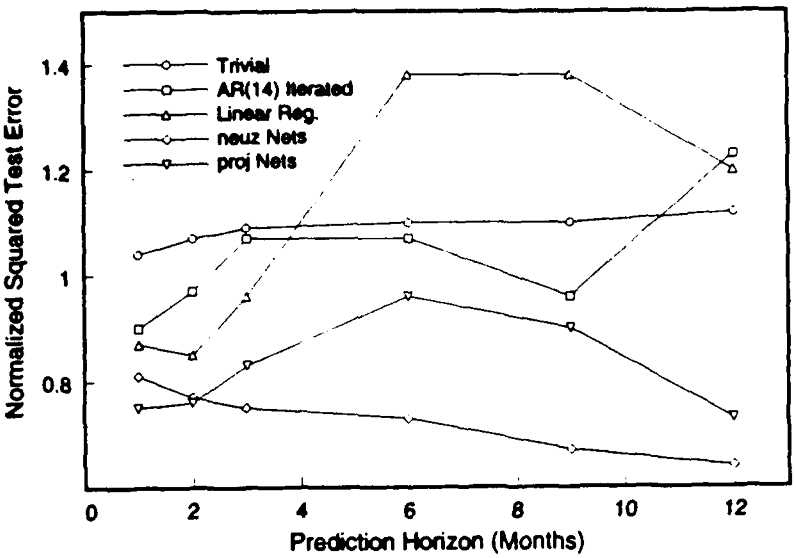
General Issues regarding Predicting Industrial Production:

* Poor Performance in GNP Forecasting by Professional Economists

* Convincing evidence for:
  non-normality in I.P
  non-stationarity in I.P

* Can we:
  find convincing evidence for non-~~stationarity~~ linearity in I.P ?
  find a best model given the "Noise - Non-stationarity Trade-off"?

Neural Nets Out-Perform Linear Models



Note: Relative Performance of NNs improves w/ Forecast Horizon ⟹ Increasing Nonlinearity w/ Horizon

Training Set Squared Error	0.7632
One Month Prediction Error	0.9039
Two Month Prediction Error	0.9733
Three Month Prediction Error	1.071
Six Month Prediction Error	1.072
Nine Month Prediction Error	0.9599
Twelve Month Prediction Error	1.229

Table 2: Results of univariate AR(14) Model for IP.R1, one month rates of return for IP. Training set error and error for iterated predictions are shown.

Components of Leading Index

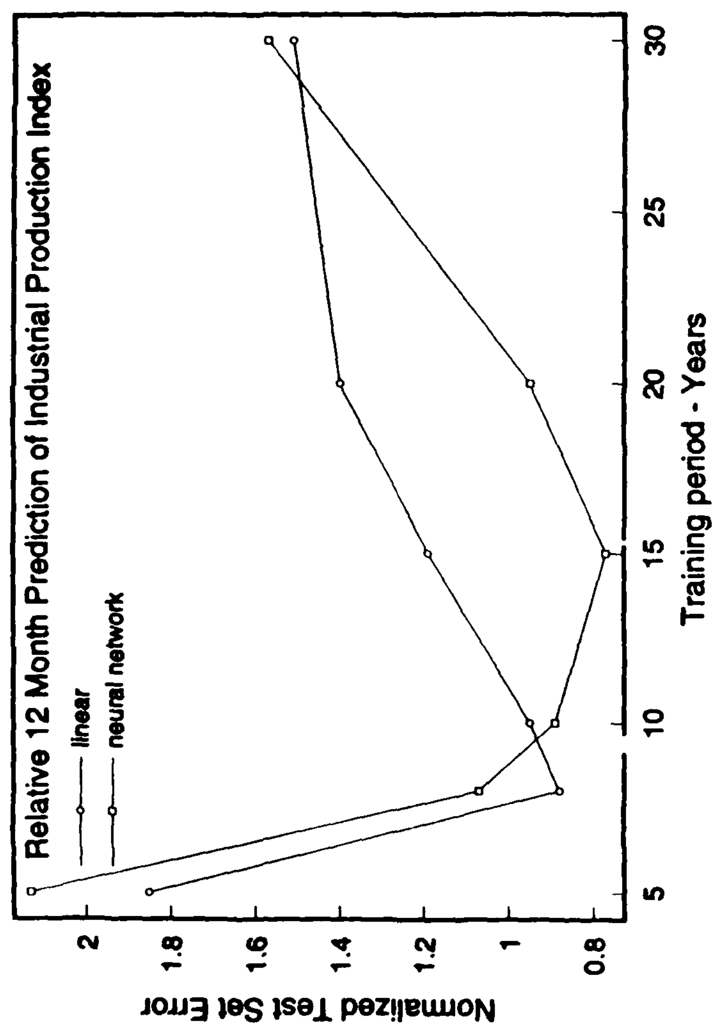
Symbol	Description
DLEAD	COMPOSITE INDEX OF 11 LEADING INDICATORS(82=100,SA)
LPHRM	AVG. WEEKLY HRS. OF PRODUCTION WKRS.: MANUFACTURING (SA)
LUINC	AVG WKLY INITIAL CLAIMS,STATE UNEMPLOY.INS.,EXC P.RICO(THOUS;SA)
MOCM82	2 MFG NEW ORDERS: CONSUMER GOODS & MATERIAL,828(BIL$,SA)
MPCON8	CONTRACTS & ORDERS FOR PLANT & EQUIPMENT IN 828(BIL$,SA)
HSBP	HOUSING AUTHORIZED: INDEX OF NEW PRIV HOUSING UNITS (1967=100;SA)
IVPAC	VENDOR PERFORMANCE: 0F CO'S REPORTING SLOWER DELIVERIES(,NSA)
HHSNTN	U. OF MICH. INDEX OF CONSUMER EXPECTATIONS(BCD-83)
IVCUD8	CHANGE MFG UNFIL.ORDER DUR.GODS,SMOOTHED(BILL.828)(BCD-92)
P3M99S	CHANGE IN SENSITIVE MATERIALS PRICES SMOOTHED DATA()BCD99
FM2D82	MONEY STOCK: M-2 IN 19828(BIL$,SA)(BCD 106)
FSPCOM	S&P'S COMMON STOCK PRICE INDEX: COMPOSITE (1941-43=10)

Table 3: Symbols and descriptions for the Index of Leading Indicators and its 11 current components.

AR Order (AIC)	Training Set	Test Set
3	1950 to 1979.92	1980 to 1990
Component	**Training Error**	**Test Error**
DLEAD.PR1	[illegible]	0.7879
IP.PR1	0.7189	0.8481

Table 4: Results of bivariate AR Model for one month rates of return for DLEAD and IP.

Evidence for Nonstationarity
the "Noise - Nonstationarity Trade-off"

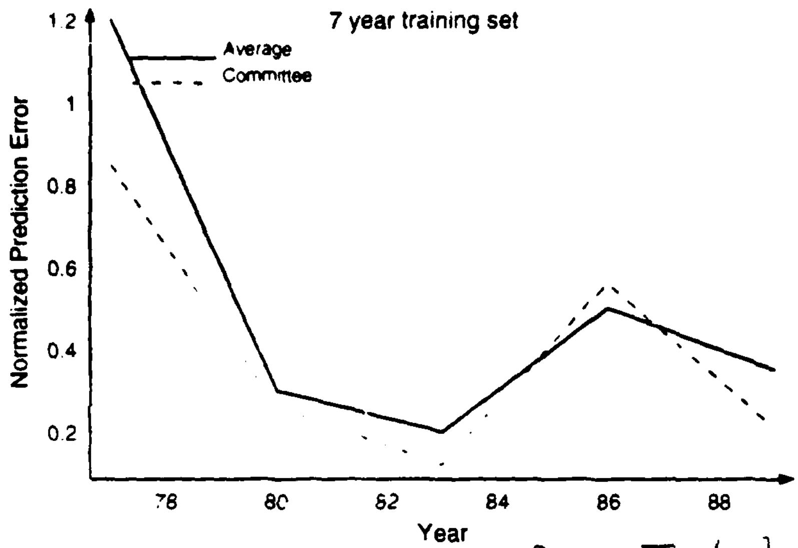


References on Combinations of Forecasts

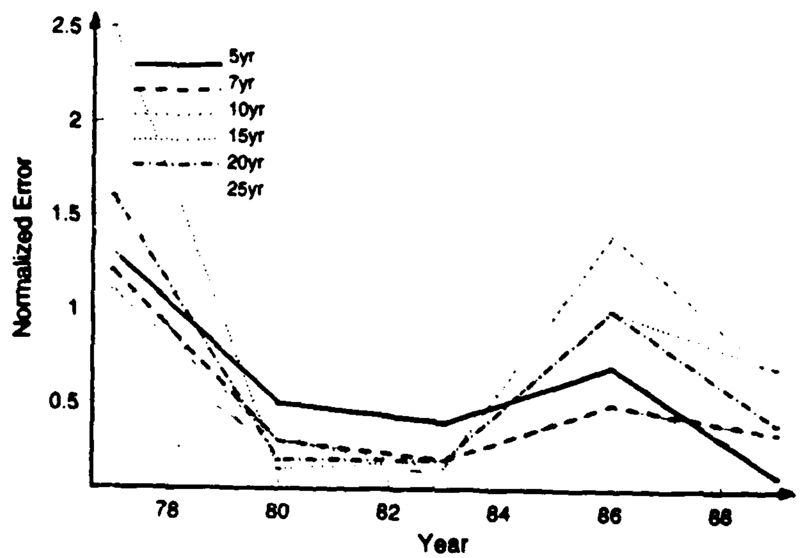
- Bates & Granger 1969
  "The Combination of Forecasts"
  Op. Res. Q <u>20</u> 451-468.
- Newbold & Granger 1974
  "Experience with ..."
  J. Roy. Stat. Soc. <u>A</u> 13<u>7</u>, 131-146
- Winkler & Makridakis 1983
  "The Combination of Forecasts"
  J. Roy. Stat. Soc. <u>A</u>146, 150-157

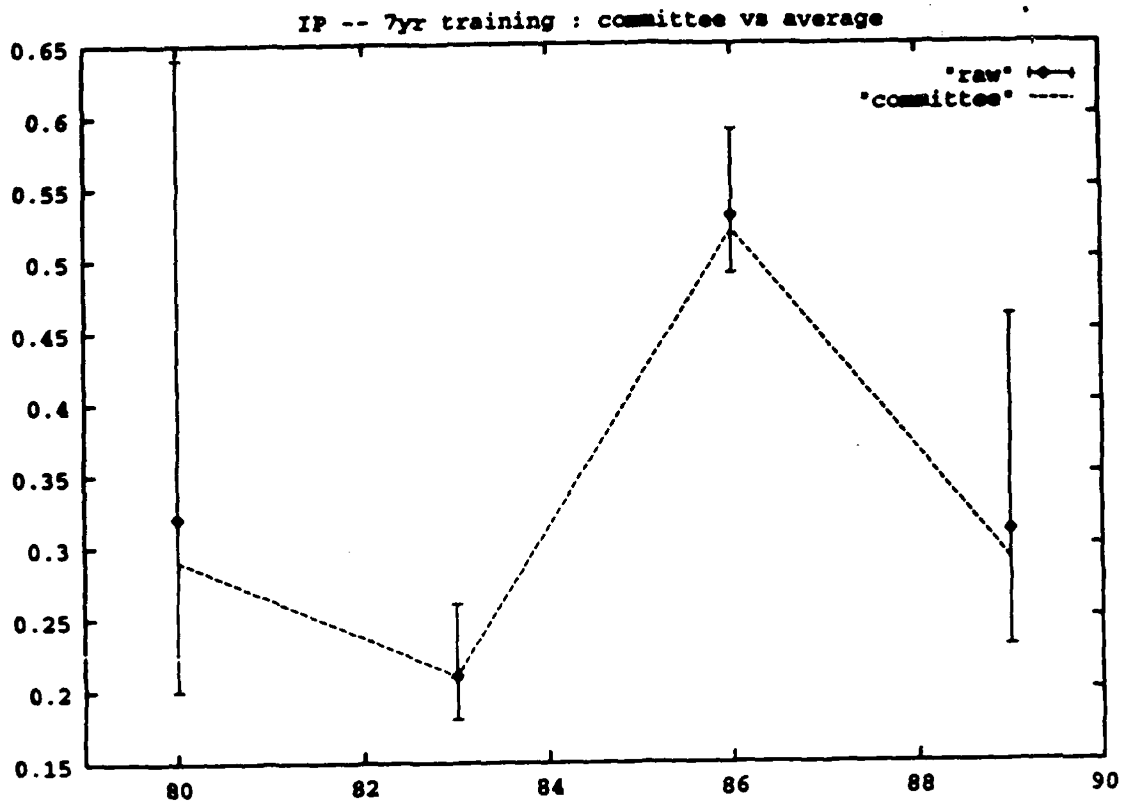
<u>Special issues of:</u>
Journal of Forecasting 1989
Int'l. J. of Forecasting 1989

<u>Book:</u> Granger & Newbold 1986
<u>Forecasting Economic Time Series</u>, Ch. 9.
Academic Press



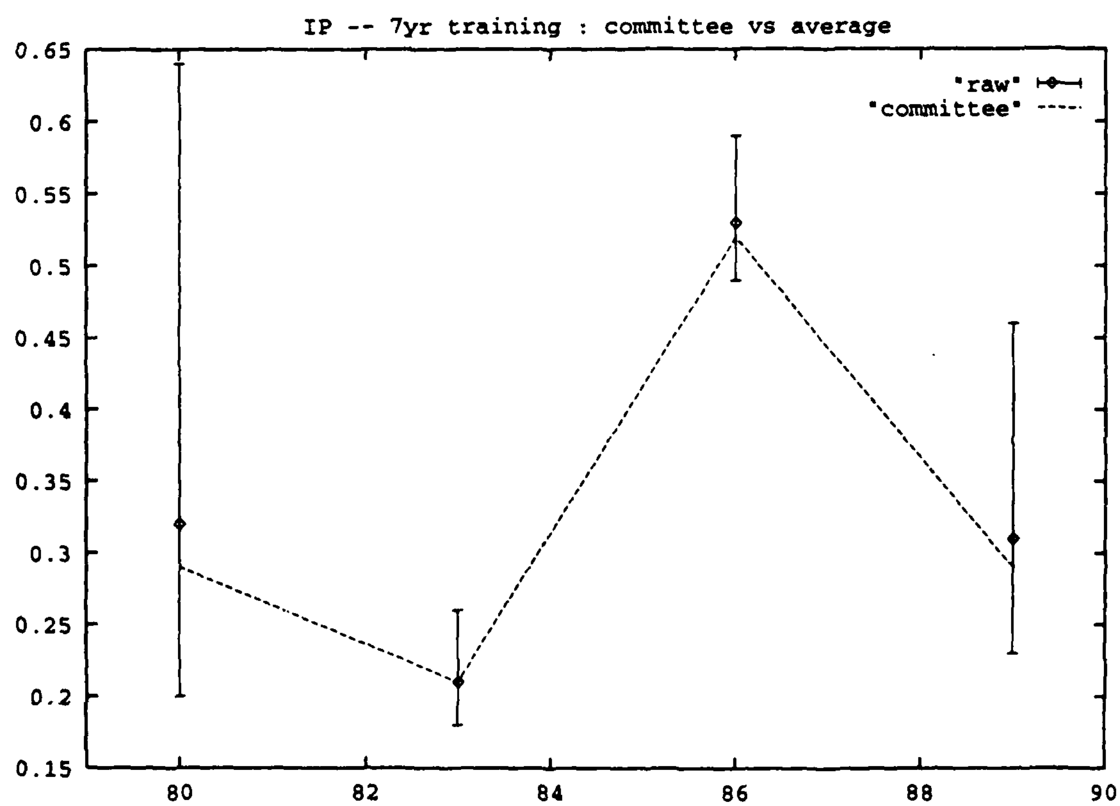
Committees outperform average performance of Individual Models





Performance of Committees is better on average and more consistent than individual models.

IP45 - new2



Overview of combining models

Wray BUNTINE

RIACS/NASA Ames Res. Ctr.

Summary of Kwok & Carter '88.

(undergraduate thesis at Sydney)

method averaging multiple decision trees: generate m trees by perturbing the splitting rule then combine their predictions

results

- ☐ averaging better than voting
- ☐ more models (larger m) better initially, but with m too large it can degrade (e.g. > 20) (presumably due to other generation method)
- ☐ averaging produces much better class probability estimates (measured by MSE / half-Brier score), whereas accuracy improvement sometimes only marginal (≈1%)

Basic Formulation of learning

model M with real-valued parameters θ

cost function, $cost(M,\theta)$, usually

$$\text{observed-error}(M,\theta) + \text{complexity}(M,\theta)$$

choose (M,θ) to minimize cost

- complexity can be regularizer, $-\log(\text{Bayesian prior})$, encoding cost, ...
- what if model family is poor?
  different model may match problem better, so give improved learning
- what if your have small data set
  many models will be near minimum, so seems unfair to pick just one
  e.g. Wedding investments

Mixture modelling

$$p(\text{resp.} \mid M,\theta,\text{input}) = \sum_c p(c \mid M,\theta,\text{input})\, p(\text{resp.} \mid c, M,\theta,\text{input})$$

gating fn. experts prediction

- i.e. extend the model to be a mixture of experts ←
- averaging is part of the network so occurs inside the cost function
- Theory says if the mixture is a better model of problem (e.g. Nowlan's speech task), then asymptotic convergence should be faster
  (i.e. this is a very loose statement!
  i.e. "better" model needs less parameters)

Model Averaging

pick multiple models M_1, θ_1, M_2, θ_2, $\ldots M_m, \theta_m$ — chosen somehow with the basic cost minimization scheme then average their results.

$$p(resp \mid Data, input) = \sum_{i=1}^{m} p(resp \mid M_i, \theta_i, input)\, w_i$$

- ☐ averaging occurs <u>outside</u> the cost function
- ☐ rationale is we aren't convinced that global cost minimum gives absolute "best" model since many others are almost as good
- ☐ should <u>do better for small samples</u> and <u>converge to non-averaging asymptotically</u>

Bayesian Averaging

(rationale for model averaging)

$$p(\text{new-response} \mid \text{Data}, \text{newinput})$$

$$= \int_{M,\theta} p(\text{new-response} \mid M, \theta, \text{newinput})\, p(M, \theta \mid \text{data})$$

model prediction — model quality

$$\cong \sum_{(M_i, \theta_i)} p(\text{new-resp} \mid M_i, \theta_i, \text{newinput})\, w(M_i, \theta_i)$$

weights for sampling scheme

i.e. use Monte Carlo to approximate the integral with finite sample of models $(M_1, \theta_1), \ldots, (M_m, \theta_m)$

e.g. Gibbs, importance sampling, branch & bound, some 'closed' form approximations

e.g. Neal, BUGS (Spiegelhalter et al.), Buntine '91, etc.

DISCUSSION

GEM
ensemble methods
committees
C&W

model averaging
i.e. better on small samples!

hierarchical mixture of experts

mixture modelling

boosting ? mixture modelling [illegible]

since each network trained on small subset
+
since boosting poor on small [illegible]

computation trick from Quinlan for improving speed of training by [illegible] on parts [illegible]

Inferring a function

vs.

inferring an inference algorithm

David Wolpert
SFI, TXN

OVERVIEW

1) $\exists$?illions of games to play w/ partitions of the training set.

2) Alas, can't prove nuthin'.

3) Theory suggests though – that inferring an inference algorithm is highly analogous to inferring an i/o function.

4) Exploit this analogy.

INFERRING INFERENCE ALGORITHMS

1) Model selection w/o data partitioning
   (E.g., AIC, GPE, ML-II)

2) Model combining w/o data partitioning
   (E.g., Perrone, Nowlan, Buntine)

3) Model selection w/ data partitioning
   (cross-validation, stratification)

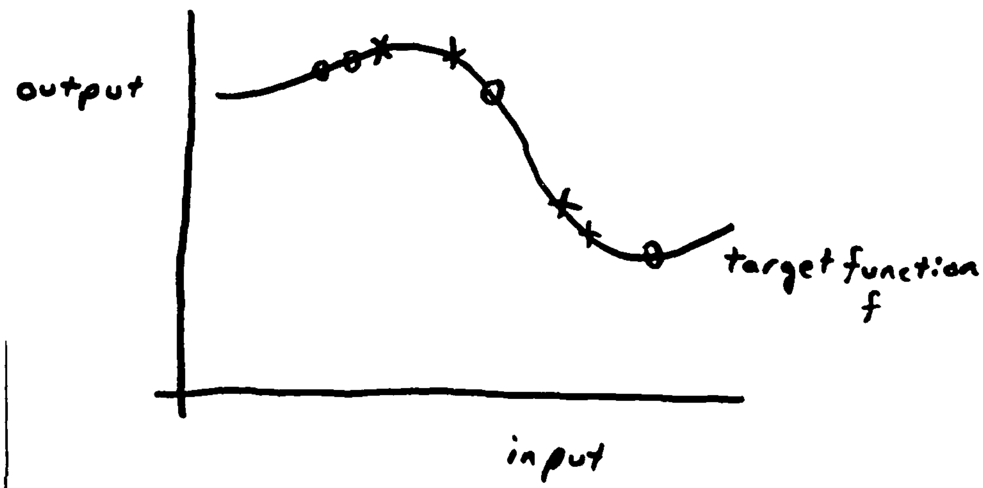
4) Model combining w/ data partitioning
   Stacking: Breiman, Friedman, Waltz,...

Tons to do w/ (3) & (4).

For example...

STRONG CROSS-VALIDATION

Illustrate for noise-free case:



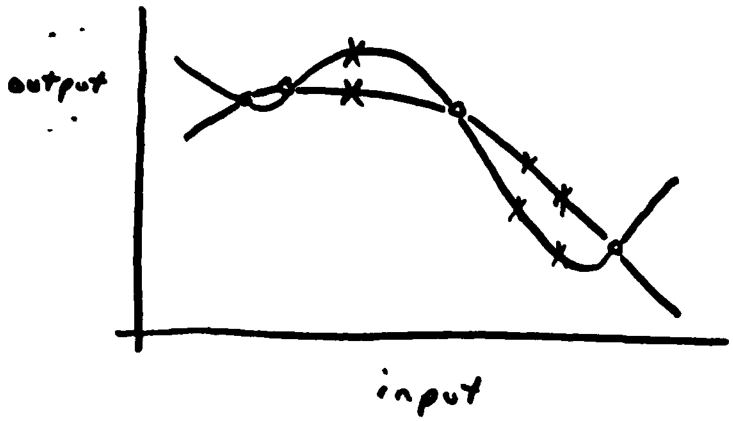
o = actual training set element

x = "alternative" training set element

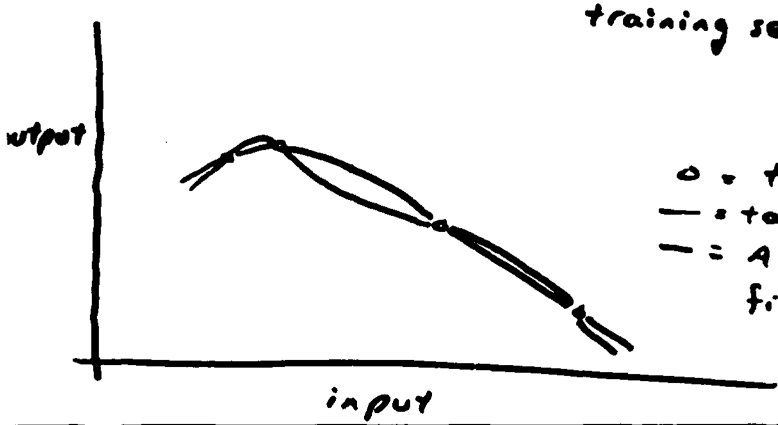
Would like small x-validation error not only within the o's, but within any set of x's lying on f.

Problem: Don't know f. But do know "proxies"...

STACKING CROSS-VALIDATION – continued



o = original training set element
— = generalizer A's fit
x = " " alternative training set element
— = generalizer B's fit
x = " " alternative training set element



o = training set
— = target function
— = A generalizer's fit

THEORY – OFF-TRAINING SET ERROR

1) No credit for memorization
(can affect "learning curve" theoretical results drastically.)

2) Life often is not iid between testing & training.

3) The whole notion of "generalize" to new examples.

So it's worth considering off-training test error (i.e. test set disjoint from training set) as well as conventional test error (i.e. test set can overlap training set).

THEORY; you can't prove anything

- f is a target function, and m the size of L.
- $E_i(.)$ refers to an expectation value using generalizer i.

For any two generalizers, independent of the noise and sampling distribution,

i) Averaged over all f,

$$E_1(E \mid f, m) - E_2(E \mid f, m) = 0;$$

ii) Averaged over all f, for any L,

$$E_1(E \mid f, L) - E_2(E \mid f, L) = 0;$$

iii) Averaged over all $P(f)$,

$$E_1(E \mid m) - E_2(E \mid m) = 0;$$

iv) Averaged over all $P(f)$, for any L,

$$E_1(E \mid L) - E_2(E \mid L) = 0.$$

THEORY; gotta make an assumption

1)

$$P(E \mid m) = \Sigma_{h,f}\, P(h \mid L)\, P(f \mid L)\, M_E(h, f)$$

For no noise, an inner product.
So must guess $P(f \mid L)$. Empirical Bayes then says ... cross-validate!

2)

$$P(E \mid s, L) = \Sigma_{h,f}\, P(h \mid L)\, P(f \mid L)\, N_{s,E,L}(h, f)$$

So "over-training" need have nothing to do with "training on the noise";
it can occur when there *is no* noise.

META-GENERALIZATION

A new event space. Define X' from X and Y:

$x' \in X'$ is an ordered set

$\{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m); q\}$.

Just as X and Y gave

f's (mappings from X to Y);

h's (mappings from X to Y);

L's (sets of X-Y pairs)

obeying $P(h, f \mid L) = P(h \mid L) \times P(f \mid L)$,

X' and Y give

d's (mappings from X' to Y);

g's (mappings from X' to Y);

ω's (sets of X'-Y pairs)

obeying $P(d, g \mid \omega) = P(g \mid \omega) \times P(d \mid \omega)$.

META-GENERALIZATION - continued

If $P(g \mid \omega)$ is the rule "choose the g best fitting ω"

- analogous to "choose the h best fitting L"-

then you have leave-one-out cross-validation.

$$P(E \mid m) = \Sigma_{g,d}\, P(g \mid \omega)\, P(d \mid \omega)\, M'_E(g, d)$$

$$P(E \mid s, \omega) = \Sigma_{d,g}\, P(g \mid \omega)\, P(d \mid \omega)\, N'_{s,E,\omega}(g, d)$$

So:

Over-cross-validating, just like over-training;

Regularized cross-validation;

Combining generalizers - stacking - just like combining hypotheses.

Theory gives *insight*, not answers.

A Bayesian Perspective on Committees

Hans H. Thodberg, Danish Meat Res. Inst.
email thodberg@nn.meatre.dk

Summary of Nips'93 poster

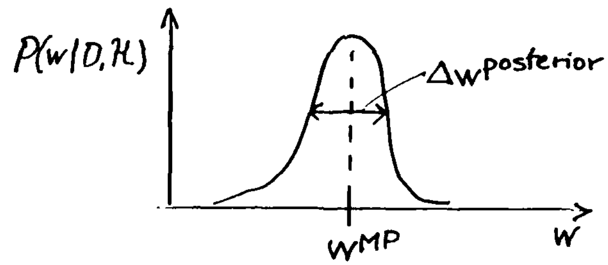
MacKay's Bayesian framework for Backpropagation is applied and extended.

- Adjusts weight decay parameters on–the–fly
- No test set is needed
- The evidence for a model = Quality measure.
  High evidence implies good generalisation
- The ev. ence is an efficient stop criterion for pruning
- The evidence of a committee is computed
- Bayesian error bars on predictions are computed
- **Real-World Application** to analytic chemistry:
  The determination of fat in minced meat by near–infrared spectroscopy.

Bayes Gives the Answer

$$P(\mathcal{H}|D) = \frac{\underbrace{\boxed{P(D|\mathcal{H})}}_{\text{the evidence for } \mathcal{H}} P(\mathcal{H})}{P(D)}$$

$$P(w|D,\mathcal{H}) = \frac{P(D|\mathcal{H},w)P(w|\mathcal{H})}{P(D|\mathcal{H})}$$



$$P(D|\mathcal{H}) = \int P(D|\mathcal{H},w)P(w|\mathcal{H})dw$$
$$P(w|\mathcal{H}) = 1/\Delta w^{prior}$$

$$\begin{aligned}
\mathrm{Ev}(\mathcal{H}) \equiv P(D|\mathcal{H}) &= P(D|\mathcal{H},w^{MP})\frac{\Delta w^{posterior}}{\Delta w^{prior}} \\
\text{Evidence} &= \text{Likelihood} \times \text{OckhamFactor} \\
\text{ModelQuality} &= \text{DataFit} \times \text{Simplicity}
\end{aligned}$$

The Four Levels of Inference

Level 1 Make predictions including error bars for new input data.

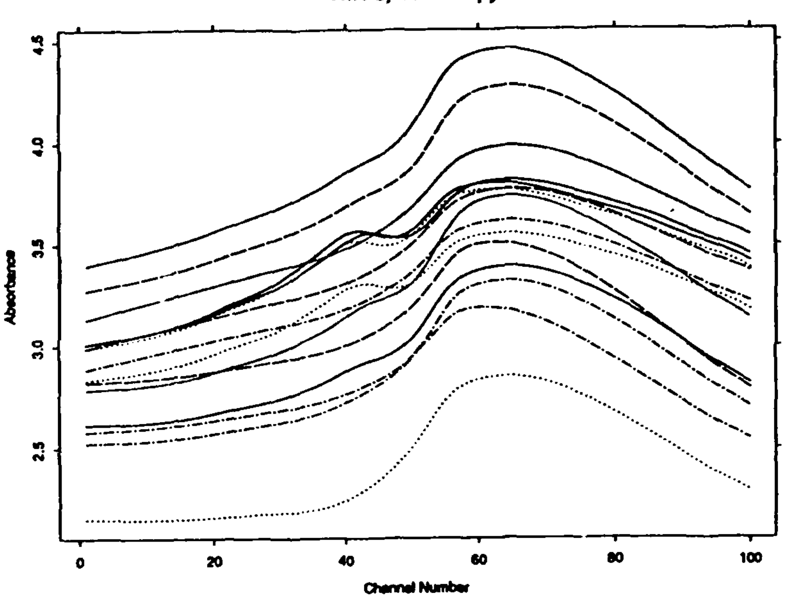
Level 2 Estimate the weight parameters and their uncertainties.

Level 3 Estimate the scale parameters and their uncertainties

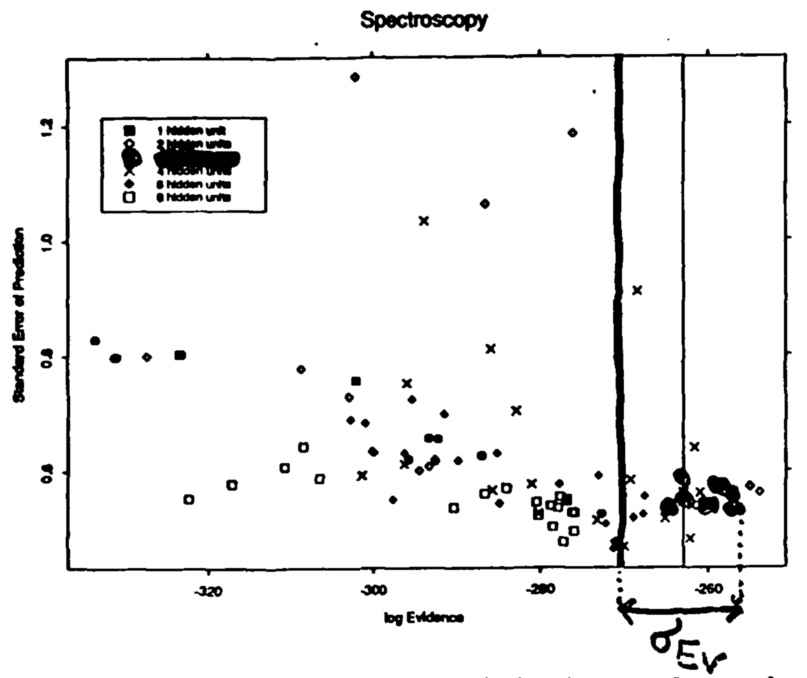
Level 4 Select the **w**-minimum and the network architecture. Optionally select a committee to reflect the uncertainty on this level.

Application to Spectroscopic Data

A Tecator near-infrared spectrometer is used to measure the light transmitted through a sample of ground meat. The purpose is to determine the fat content, whci varies between 5 and 50%.



Fifteen NIR-transmission spectra



The test error as a function of the log evidence for networks trained on the spectroscopic data. Networks with 2 and 3 hidden units attain the highest evidence. The 20 networks with highest evidence are delimited by the vertical line and have on average SEP=0.55.

The Evidence for a Committee of Networks

Train several networks with different numbers of hidden units and different initial weights.

Select a committee consisting of the networks with the best evidence within an estimated uncertainty

- The average of the committee gives a better generalisation than the average network in the committee.
- The degree of dissent within the committee is used to compute the uncertainty of the prediction.

N_C is the number of different solutions in the committee (same architecture).

Ock($\mathbf{w}$) is N_C times smaller:

$$\log \mathrm{Ev}(C) = \log N_C + \log \mathrm{Ev}(\mathcal{H})$$

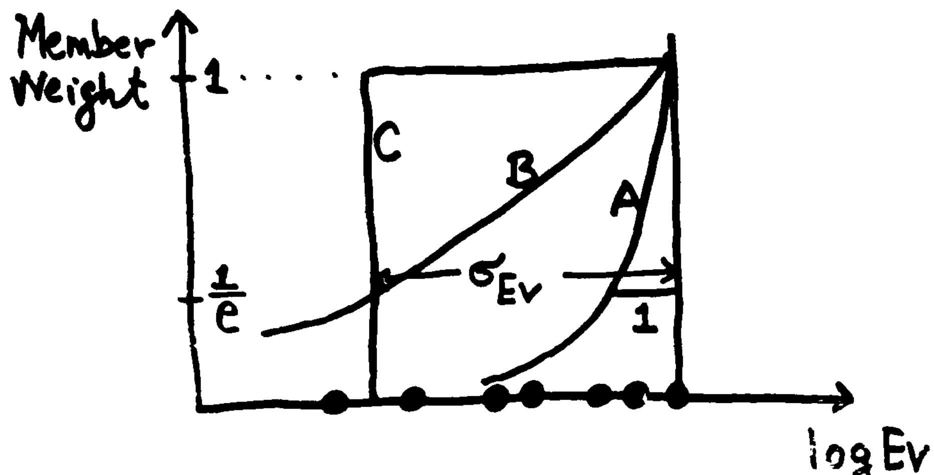
How to weight the members of the committee?

→ according to the evidence $Ev(\mathcal{H}) = p(D|\mathcal{H})$

However...

$$\log Ev_{GaussApp}(\mathcal{H}) = \log Ev(\mathcal{H}) + \log \Delta Ev$$

$\log \Delta Ev \sim N(0, \sigma_{Ev}^2)$



I use MemberWeight C as an approximation to B

The committee prediction

$$y(\mathbf{x}) = \frac{1}{N_C} \sum_{j=1}^{N_C} y(\mathbf{x}, \mathbf{w}_{\mathrm{MP}}^j)$$

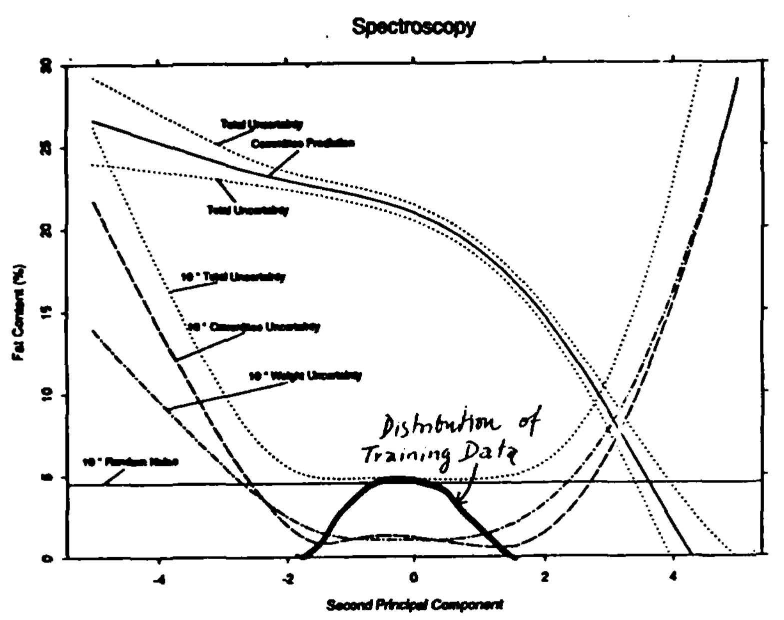
The committee members disagree to a certain extent and this committee uncertainty (CU) gives the following contribution to the prediction variance:

$$\sigma_{cu}(\mathbf{x})^2 = \frac{1}{N_C - 1} \sum_{j=1}^{N_C} (y(\mathbf{x}) - y(\mathbf{x}, \mathbf{w}_{\mathrm{MP}}^j))^2$$

The total prediction variance is

$$\sigma_{total}(\mathbf{x})^2 = \sigma_\nu^2 + \sigma_{wu}(\mathbf{x})^2 + \sigma_{cu}(\mathbf{x})^2$$

See the figure!



Prediction of the fat content as a function of the second principal component p_2 of the NIR spectrum. 95% of the training data has $|p_2| < 2$. The total standard error bars are indicated by a "1 sigma" band (dotted lines). The total standard errors and the standard errors of its contributions are shown separately, multiplied by a factor of 10.

Conclusions

- Bayes gives a rational account for committees

  (Ockham does not! – committees are not "simple")

- Realistic Error Bars require Committee Uncertainty term

- Principle for Active Learning:

  Ask if the experts disagree

References

- H.H.Thodberg, "A Review of Bayesian Neural Networks with an Application to Near Infrared Spectroscopy" and "A Bayesian Approach to Pruning of Neural Networks" submitted to *IEEE Trans. of Neural Networks* 1993. (in thodberg.ace-of-bayes.ps.Z in the neuroprose electronic archive at Ohio)
- D.J.C.MacKay, "A Practical Bayesian Framework for Backpropagation Networks" *Neural Comp.* 4 (1992) 448-472.
- H.H.Thodberg, "Improving Generalization of Neural Networks through Pruning", *Int.J.Neural Systems* 1 (1991) 317-325.
- C.Bishop, "Exact Calculation of the Hessian Matrix for the Multilayer Perceptron" *Neural Comp.* 4 (1992) 494-501.
- C.Borggaard and H.H.Thodberg, "Optimal Minimal Neural Interpretation of Spectra", *Analytic Chemistry* 64 (1992) 545-551.

MERITS OF COMBINING NEURAL NETWORKS
POTENTIAL BENEFITS AND RISKS

SHERIF HASHEM*[1]

SCHOOL OF INDUSTRIAL ENGINEERING
PURDUE UNIVERSITY

DECEMBER 4, 1993

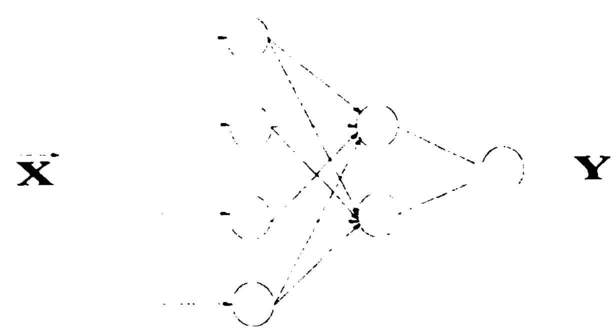
[1]*Supported by PRF Research Grant 6901627 from Purdue University, West Lafayette, IN

OUTLINE

- Optimal Linear Combinations of Neural Networks.
  - Motivation.
  - Definition.
  - Combination weights.
- Benefits of Combining:
  - For well-trained networks.
  - For poorly trained networks.
- Ill Effects of Collinearity:
  - Computational ill effects.
  - Statistical ill effects.
- Concluding Remarks.

OPTIMAL LINEAR COMBINATIONS OF NEURAL NETWORKS

MOTIVATION



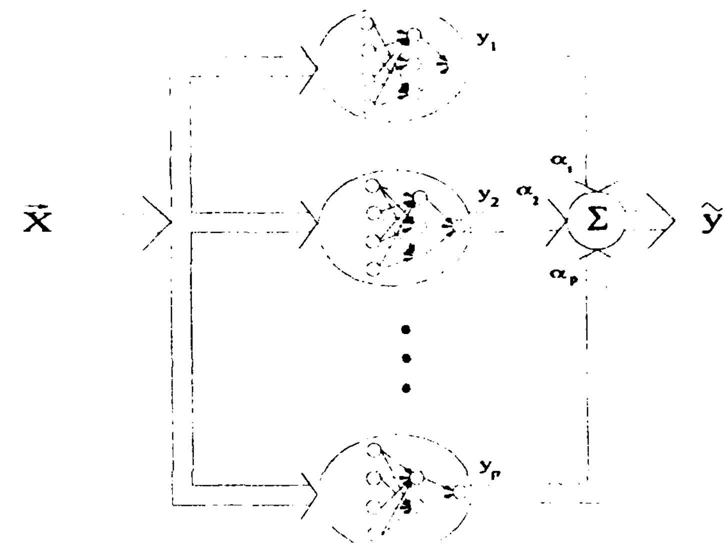
Neural Network Model

$$Y = f_{NN}(\vec{X})$$

Problem: Given a (process) data set, construct a Neural Network based model that "closely" approximates the underlying process.

OPTIMAL LINEAR COMBINATIONS OF NEURAL NETWORKS

DEFINITION



$$\tilde{Y} = \sum_{i=1}^{p} \alpha_i Y_i = \vec{\alpha}^t \vec{Y}.$$

- **Current approach:** Train many NNs, then pick the "best."
- **New approach:** Optimal Linear Combinations (OLC) of NNs.

COMBINATION-WEIGHTS

Optimality Criterion:

Minimize the Mean Squared Error (MSE) over observed data.

$$\mathrm{MSE} = \mathrm{E}_{\vec{X}}(r(\vec{X}) - \tilde{Y})^2.$$

The MSE-optimal weights:

$$\vec{\alpha}^* = \Phi^{-1}\vec{\Theta},$$

where

$\Phi = [\phi_{ij}] = [\mathrm{E}(y_i(\vec{X})\, y_j(\vec{X}))]_{p\times p}$ and $\vec{\Theta} = [\theta_i] = [\mathrm{E}(r(\vec{X})\, y_i(\vec{X}))]_{p\times 1}$.

In practice: Given a data set $\mathcal{D}$, estimate $\vec{\alpha}^*$ using

$$\hat{\phi}_{ij} = \sum_{k=1}^{|\mathcal{D}|}(y_i(x_k)\, y_j(x_k))/|\mathcal{D}| \quad \forall\, i,j\,;$$

$$\hat{\theta}_i = \sum_{k=1}^{|\mathcal{D}|}(r(x_k)\, y_i(x_k))/|\mathcal{D}| \quad \forall\, i\,.$$

OTHER FORMS OF MSE-OLC

A. Unconstrained MSE-OLC with a constant term.

$$\tilde{Y} = \sum_{i=0}^{p} \alpha_i Y_i = \vec{\alpha}'\vec{Y}.$$

B. Constrained MSE-OLC with a constant term.

$$\tilde{Y} = \sum_{i=0}^{p} \alpha_i Y_i = \vec{\alpha}'\vec{Y}, \qquad \sum_{i=1}^{p} \alpha_i = 1.$$

C. Constrained MSE-OLC without a constant term.

$$\tilde{Y} = \sum_{i=1}^{p} \alpha_i Y_i = \vec{\alpha}'\vec{Y}, \qquad \sum_{i=1}^{p} \alpha_i = 1.$$

D. Convex MSE-OLCs:

$$\tilde{Y} = \sum_{i=1}^{p} \alpha_i Y_i = \vec{\alpha}'\vec{Y}, \qquad \sum_{i=1}^{p} \alpha_i = 1, \qquad 1 \ge \alpha_i \ge 0.$$

BENEFITS OF COMBINING

FOR WELL-TRAINED NETWORKS

EXAMPLE 1

- **Problem:** Consider approximating

$$r(X) = 0.02(12+3X-3.5X^2+7.2X^3)(1+\cos 4\pi X)(1+0.8\sin 3\pi X)$$

where $X \in [0,1]$.

- **NN model:**
  - *Topology:* Three 1-5-5-1 & three 1-10-1 NNs
  - ***Training algorithm:*** Error backpropagation
  - ***Training data:*** 200 uniformly distributed points $(x_i, r(x_i))$
  - ***MSE-OLC fitting data:*** same 200 points

- **Resultant "true [1]" MSE:**

Best NN (NN4): 0.000137.

Simple averaging: 0.000396.

MSE-OLC: 0.000017;

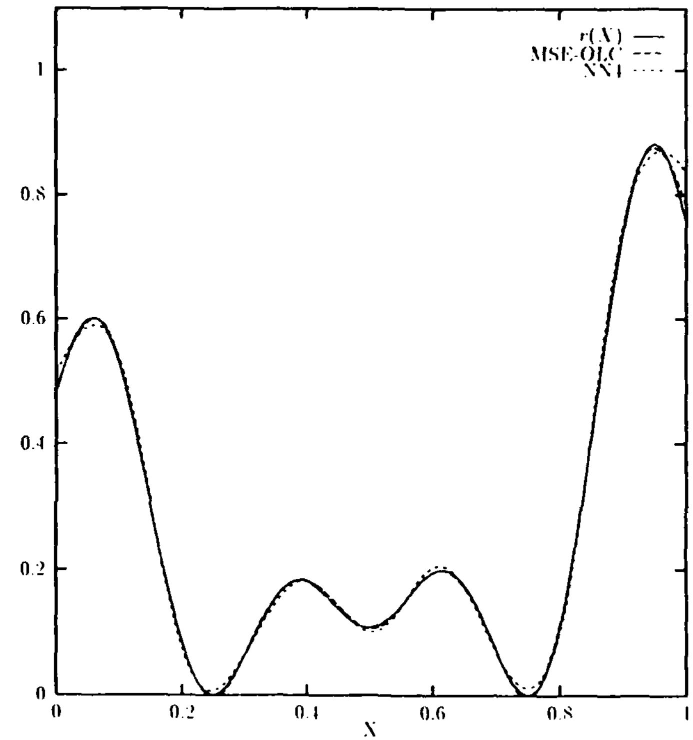
88 % less than NN4; **96 %** less than simple averaging

[1] Computed on the (true) known function, $r(X)$

BENEFITS OF COMBINING

FOR WELL-TRAINED NETWORKS

EXAMPLE 1 (Cont.)



BENEFITS OF COMBINING

FOR WELL-TRAINED NETWORKS

EXAMPLE 2

- **Problem:** Consider approximating

$$r(X) = \sin[2\pi(1-X)^2], \quad \text{where } X \in [0,1].$$

- **NN model:**
  - *Topology:* Two 1-3-1, two 1-2-2-1, & two 1-4-1 NNs.
  - ***Training algorithm:*** Using Error Backprop for 5000 iterations.
  - ***Training data:*** 10 uniformly distributed data points.
  - ***MSE-OLC fitting data:*** same 10 points.

- **Resultant "true" MSE:**

Best NN (NN6): 0.044.

Simple averaging: 0.072.

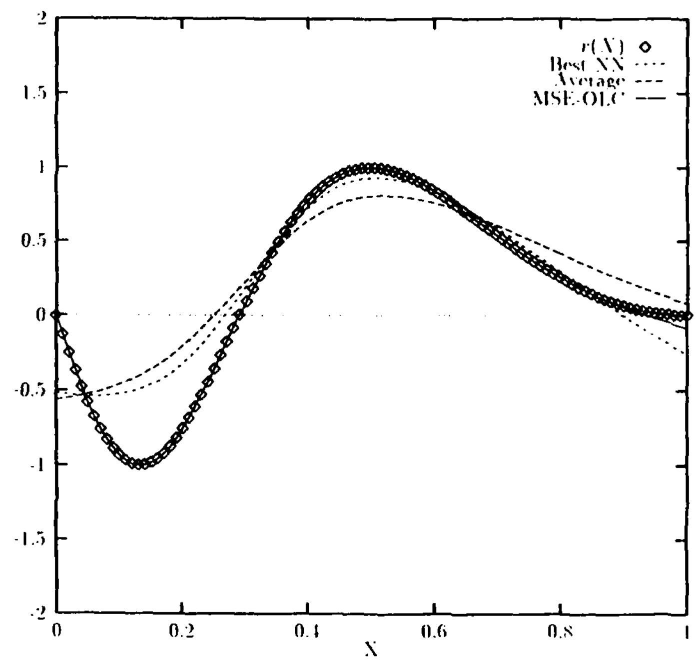
MSE-OLC: 0.00020:

99+ % less than NN6 or simple averaging.

BENEFITS OF COMBINING

FOR WELL-TRAINED NETWORKS

EXAMPLE 2 (Cont.)



BENEFITS OF COMBINING

FOR POORLY TRAINED NETWORKS

EXAMPLE 3

- **Problem:** Consider approximating

$$r(X) = \sin[2\pi(1-X)^2], \quad \text{where} X \in [0,1].$$

- **NN model:**
  - *Topology:* Two 1 3 1, two 1 2 2 1, & two 1 4 1 NNs.
  - *Training algorithm:* Using Error Backprop for 2000 iterations.
  - *Training data:* 10 uniformly distributed data points.
  - *MSE-OLC fitting data:* same 10 points.

- **Resultant "true" MSE:**

Best NN (NN6): 0.219.

Simple averaging: 0.241.

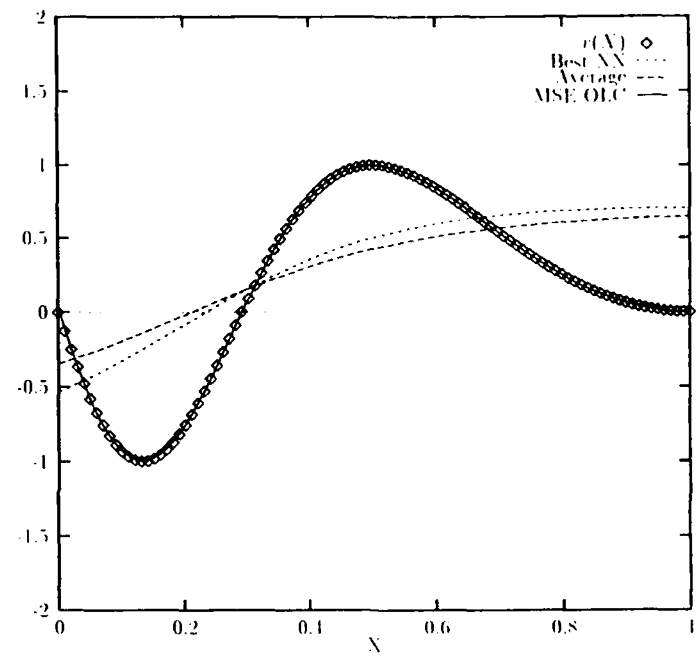
MSE-OLC: 0.000060:

99+ % less than NN6 or simple averaging.

BENEFITS OF COMBINING

FOR POORLY TRAINED NETWORKS

EXAMPLE 3 (Cont.)



ILL EFFECTS OF COLLINEARITY

MSE-OLCs:

$$\tilde{Y} = \sum_{i=1}^{p} \alpha_i Y_i = \vec{\alpha}^t \vec{Y}.$$

Unconstrained MSE-OLC Weights:

$$\vec{\alpha}^* = \Phi^{-1}\vec{\Theta},$$

where

$\Phi = [\phi_{ij}] = [\mathrm{E}(y_i(\vec{X})y_j(\vec{X}))]_{p\times p}$ and $\vec{\Theta} = [\theta_i] = [\mathrm{E}(r(\vec{X})y_i(\vec{X}))]_{p\times 1}$.

Constrained MSE-OLC Weights:

$$\vec{\alpha}^* = \Omega^{-1}\vec{1} / (\vec{1}^t\,\Omega^{-1}\,\vec{1}),$$

where $\Omega = [\omega_{ij}] = \left[\mathrm{E}\left(\delta_i(\vec{X})\,\delta_j(\vec{X})\right)\right]$ is a $p \times p$ matrix, and $\vec{1}$ is a $p \times 1$ vector with all components equal to one.

Computational Ill Effects:

Near singular matrices (inversion, sensitivity, round-off errors).

Statistical Ill Effects:

Collinearity can undermine the robustness (generalization ability) of the MSE-OLC.

CONCLUDING REMARKS

- MSE-OLCs can significantly improve model accuracy.
- The effectiveness of MSE-OLC is not dependent on the accuracy of the component networks.
- MSE-OLC is straightforward and requires modest computational effort.
- MSE-OLC can be used to create hybrid models containing non-neural network components.

REFERENCES

R. T. Clemen. Combining forecasts: A review and annotated bibliography. ***International Journal of Forecasting***, 5:559–583, 1989.

D. A. Belsley. *Conditioning Diagnostics: Collinearity and Weak Data in Regression*. John Wiley & Sons, New York, 1991.

C. Genest and J. V. Zidek. Combining probability distributions: A critique and an annotated bibliography. *Statistical Science*, 1(1):114–148, 1986.

S. Hashem. *Optimal Linear Combinations of Neural Networks*. PhD thesis, School of Industrial Engineering, Purdue University, Dec. 1993.